Distributed Security Policy Analysis

POLITECNICO DI TORINO

SCUOLA DI DOTTORATO

Dottorato in Ingegneria Informatica e dei Sistemi – XXVIII ciclo

Tesi di Dottorato

Distributed Security Policy Analysis

The complete solution

Christian PITSCHEIDER

Tutore
Prof. Antonio Lioy

Coordinatore del corso di dottorato
prof. Matteo Sonza Reorda

Marzo 2016

Summary

Computer networks have become an important part of modern society, and computer network security is crucial for their correct and continuous operation. The security aspects of computer networks are defined by network security policies. The term policy, in general, is defined as "a definite goal, course or method of action to guide and determine present and future decisions" [1]. In the context of computer networks, a policy is "a set of rules to administer, manage, and control access to network resources" [2]. Network security policies are enforced by special network appliances, so called security controls. Different types of security policies are enforced by different types of security controls.

Network security policies are hard to manage, and errors are quite common. The problem exists because network administrators do not have a good overview of the network, the defined policies and the interaction between them. Researchers have proposed different techniques for network security policy analysis, which aim to identify errors within policies so that administrators can correct them. There are three different solution approaches: anomaly analysis, reachability analysis and policy comparison. *Anomaly analysis* searches for potential semantic errors within policy rules, and can also be used to identify possible policy optimizations. *Reachability analysis* evaluates allowed communication within a computer network and can determine if a certain host can reach a service or a set of services. *Policy comparison* compares two or more network security policies and represents the differences between them in an intuitive way. Although research in this field has been carried out for over a decade, there is still no clear answer on how to reduce policy errors. The different analysis techniques have their pros and cons, but none of them is a sufficient solution. More precisely, they are mainly complements to each other, as one analysis technique finds policy errors which remain unknown to another. Therefore, to be able to have a complete analysis of the computer network, multiple models must be instantiated.

An analysis model that can perform all types of analysis techniques is desirable and has three main advantages. Firstly, the model can cover the greatest number of possible policy errors. Secondly, the computational overhead of instantiating the model is required

only once. Thirdly, research effort is reduced because improvements and extensions to the model are applied to all three analysis types at the same time. Fourthly, new algorithms can be evaluated by comparing their performance directly to each other.

This work proposes a new analysis model which is capable of performing all three analysis techniques. Security policies and the network topology are represented by the so-called *Geometric-Model*. The Geometric-Model is a formal model based on the set theory and geometric interpretation of policy rules. Policy rules are defined according to the condition-action format: if the condition holds then the action is applied. A security policy is expressed as a set of rules, a resolution strategy which selects the action when more than one rule applies, external data used by the resolution strategy and a default action in case no rule applies.

This work also introduces the concept of Equivalent-Policy, which is calculated on the network topology and the policies involved. All analysis techniques are performed on it with a much higher performance. A precomputation phase is required for two reasons. Firstly, security policies which modify the traffic must be transformed to gain linear behaviour. Secondly, there are much fewer rules required to represent the global behaviour of a set of policies than the sum of the rules in the involved policies.

The analysis model can handle the most common security policies and is designed to be extensible for future security policy types. As already mentioned the Geometric-Model can represent all types of security policies, but the calculation of the Equivalent-Policy has some small dependencies on the details of different policy types. Therefore, the computation of the Equivalent-Policy must be tweaked to support new types. Since the model and the computation of the Equivalent-Policy was designed to be extendible, the effort required to introduce a new security policy type is minimal. The anomaly analysis can be performed on computer networks containing different security policies. The policy comparison can perform an Implementation-verification among high-level security requirements and an entire computer network containing different security policies. The policy comparison can perform a Change-impact-analysis of an entire network containing different security policies.

The proposed model is implemented in a working prototype, and a performance evaluation has been performed. The performance of the implementation is more than sufficient for real scenarios. Although the calculation of the Equivalent-Policy requires a significant amount of time, it is still manageable and is required only once. The execution of the different analysis techniques is fast, and generally the results are calculated in real time. The implementation also exposes an API for future integration in different frameworks or software packages. Based on the API, a complete tool was implemented, with a graphical user interface and additional features.

Contents

List of Tables

List of Figures

Part I

Introduction and problem statement

Chapter 1

Introduction

Computer networks have become the backbone of several services for citizens, companies and public administrations. Therefore, the correct and continual operation of these networks has become a critical factor. More and more computer networks are connected to the Internet, and remote sites are becoming more frequent. This global connectivity also has a downside, cyber attacks are becoming more frequent and are now the biggest threat to businesses worldwide. An annual study of the cost of cyber crime [3] concludes that although "activities relating to IT security in the network layer receive the highest budget allocation [...] cyber crimes continue to be on the rise for organizations". Furthermore, they state that "all industries fall victim to cybercrime, but to different degrees" and that "business disruption represents the highest external cost, followed by the costs associated with information loss".

Different studies exist about how to calculate the cost of a cyber attack. According to Patterson [4] the "estimated average cost of 1 hour of downtime = employee costs per hour * fraction employees affected by outage + Average Income per hour * Fraction income affected by outage". Gwebu et al. [5] state that "the damage of a breach most likely stems from direct costs such as compensation and litigation costs rather than indirect costs such as tarnished reputation and a decrease in market share and sales". Although Ko and Dorantes [6] confirm these findings "information security breaches have minimal longterm economic impact", cyber crimes also have an impact on the shareholders wealth. Gatzlaff and McCullough [7] conclude that "the stock market responds negatively to announcements of breaches of customer and/or employee data at publicly traded firms" and "negative reaction is stronger for firms with higher growth opportunities".

Goldstein et al. [8] define two broad categories of IT operational risk, which are based on the types of assets comprising IT systems. Data-related IT operational risk is defined as any threat to the confidentiality of data assets which can result in the disclosure, misuse,

or destruction of these assets. Function-related IT operational risk is defined as any threat to the availability or to the integrity of functional IT assets (that may eventually affect data assets). Their research examines the economic impact of these two categories and IT operational risk. They conclude that "IT operational risk events are value relevant in the sense that they impose a strong negative impact on the market values of organizations that experience such events" and that "these results are largely driven by Function events, such as Data events do not result in a significant effect on firms' market value"

The main reason for cyber attacks is not that computer networks are not protected at all, but that the attack surface is too big to handle. The main line of defence of modern computer networks is a firewall, but as some studies have shown, firewall configuration errors are quite frequent [9]. Network administrators have to configure firewalls and other security controls based on high-level business-oriented directives. According to Wool et al. [10] the enforcement of high-level directives requires the configuration of tens of security controls with hundreds of rules each.

Other reports that have investigated this area conclude that human errors are one of the biggest threats to computer security. The *Data Breach Investigations Report* from 2008 [11] concluded that "misconfiguration was the leading category of error contributing to data compromise". In recent years [12], [13] misconfiguration was still in the top 5 threat action varieties. Furthermore, the *Data Breach Investigations Report* from 2015 [13] states that "60% of incidents were attributed to errors made by system administrators - prime actors responsible for a significant volume of breaches and records". The *UK Security Breach Investigations Report* [14] states that "whilst many of the organizations investigated actually had firewalls installed, poor configuration of these devices rendered most of them useless. In over 96% of cases, requirement 1 of PCI-DSS [15] was not adequately adhered to". In general, their conclusion about PCI-DSS compliance is devastating: "The maximum number of requirements met by an individual organisation was only 6 out of 12, in approximately 4% of cases." These studies conclude that network administrators are unable to implement high-level security directives correctly. A report by Forrester Research [16] states that "The majority of organizations today have a set of comprehensive information security policies, but very few can confidently say that they enforce these policies consistently across the organization." and that "they cannot prove enforcement, or it is prohibitively expensive to do so".

Oppenheimer [17] concludes that service availability is influenced by incorrect system configurations in two ways: Firstly: "An operator performing any kind of action on a service (e.g., establishing a new configuration, adding a new back-end node, moving a component or user from one server to another, deleting files considered unnecessary, etc.)

may perform the task incorrectly if she does not understand the existing system configuration." Secondly: "When diagnosing a problem (be it operator-induced or not), an operator must understand the existing system configuration and sometimes the history of system configurations before the problem began – in order to follow cause-and-effect chains back to the problem's root cause."

Network Functions Virtualization (NFV) [18] and Software-Defined Networks (SDN) [19] are new network technologies which promote a more efficient and dynamic use of network resources. The goal of Network Functions Virtualization (NFV) is to transform network functions into software modules, named Virtual Network Functions (VNF), which can be executed on standard high-volume servers. NFV defines a virtualized infrastructure which supports a dynamic and flexible deployment of VNFs. This infrastructure has the advantage that administration tasks, response times and costs are reduced. The goal of Software-Defined Networks (SDN) is to reconfigure dynamically the network, by selecting arbitrary parts and redirecting traffic through specific middleboxes. This has the advantage that network resources are utilized more efficiently because the network traffic is processed by a minimal number of middleboxes. The combination of these two technologies enables the deployment of an elastic, dynamic and more efficient network architecture. However, these advantages come at a cost as the complexity of computer networks increases even further.

Although NFV and SDN are still research concepts, a widespread adoption can be expected in the coming years. Some commercial products based on these technologies already exist, and open-source implementations are available. Since the complexity of computer networks is expected to increase, network administrators require dedicated tools to reduce configuration errors. Over the last decade researchers have examined different approaches to reduce configuration errors. This research area is called *security policy analysis* and includes anomaly analysis, reachability analysis and policy comparison. Each of the three analysis techniques provides a part of the overall solution and has its advantages and disadvantages [20].

Anomaly analysis is designed to identify potential semantic errors between correlated policy rules. It is called Intra-Policy analysis when a single policy is analysed and Inter-Policy analysis when a set of interconnected policies are analysed. Rule anomalies are also used as a starting point for policy optimization.

Reachability analysis is designed to evaluate allowed communications within a computer network. It verifies if a certain host can reach a service or a set of services. Reachability analysis can be performed online on a deployed network or offline on an accurate representation of the network and its security policies. Online reachability analysis is generally executed by using tools such as "ping" or "traceroute", whereas offline reachability

analysis evaluates reachability queries on a model of the network.

Policy comparison has two use-cases: Change-impact-analysis and Implementation-verification. Change-impact-analysis evaluates possible side-effects introduced by a policy modification. Implementation-verification verifies that a high-level policy has been correctly deployed into the network. In both cases two or more security policies are compared to each other and differences are highlighted. The security policy involved can have different abstraction levels; it can be a single concrete security control configuration, sets of configurations, and high-level policies of an entire network.

By comparing the different solution approaches to each other, it becomes evident that there is no interoperability among the various models [20]. Furthermore, the number of supported security policy types is very limited and the performance tests executed are also very limited or inexistent [20]. Therefore, future research and practical usage will have to confront various disadvantages. The most significant disadvantage for future research is that the model of a security policy type for one solution can not be reused by another one. From a practical point of view, the biggest disadvantage is that the instantiation time of a model must be repeated for each analysis type. Another disadvantage is that it is nearly impossible to compare the different approaches based on their performance.

This work proposes to overcome these disadvantages by incorporating all three analysis types into a unified analysis model. This new analysis framework is designed to be extensible and, therefore, a solid base for future research. The main advantages of a unified analysis model are as follows: firstly, the model can cover the greatest possible error types; secondly, to execute all three analysis types the network administrator must instantiate only one model and, therefore, insert the information about the network topology and the security policies only once; thirdly, a new security policy type must be modelled only once and can then be used by all three analysis techniques; fourthly, improvements to different analysis techniques can be evaluated and compared more efficiently.

The proposed model also includes some new properties never proposed before, partially enabled by the unified design. The support for different security policy types is an essential feature of this work. Consequently, all analysis types also support a great variety of security policies. The supported security policy types are packet-filter, statefull, Layer7, Routing, IPsec, SSL, NAT/NAPT, Web-Proxies and Monitoring. This also highlights one of the main advantages of the unified model, a new security policy type is modelled only once but was accessible to three analysis types.

The anomaly analysis can be performed on computer networks containing all supported security policy types. The state-of-the-art solutions can only handle packet-filter policies and IPsec Policies. Furthermore, a new anomaly class is introduced to represent inconsistencies between different policy types.

The reachability analysis can verify reachability queries on computer networks containing all supported security policy types. The advantage over the state of the art is that now IPsec policies are also supported. Furthermore, the expressiveness of the reachability queries has been improved.

The policy comparison can be performed between high-level security requirements and an entire computer network containing different security policies. The policy comparison is also able to perform a Change-impact-analysis of an entire network containing different security policies. In comparison with the state of the art these two features are new, because past research focused mainly on single filtering policies.

The model takes the network topology as input, and the network security policies written in different formats and for different security controls. For example, the input taken could be the global network security policy, defined in a technology-independent formal language, and the complete network structure with all its concrete configurations. The model can then perform a policy comparison between the two input formats and verify that the implementation follows the desired network security policy. As a further step, network administrators can verify reachability of critical components or perform a conflict analysis to identify potential policy errors. Furthermore, the proposed model is extensible and flexible and can accommodate all types of security controls and network topologies. To support the different types of computer networks, the model can compose different security controls in a different order. Security controls are modelled so that they are completely independent of the network topology. To be prepared for future security controls, the model can include new security controls without significant changes to the model itself.

These key properties are accomplished by representing the network topology and its security policies with the *Geometric-Model* [21] and the use of a new concept called *Equivalent-Policy*. The Equivalent-Policy is like a black-box representation of the entire network and is calculated on the network topology and the policies involved. The analysis techniques are performed on the Equivalent-Policy, since it is a much more efficient representation of the network topology. The biggest advantage of the Equivalent-Policy is that it requires fewer rules to represent a complete network than the sum of all policy rules involved.

The Geometric-Model is a formal model based on the set-theory and and geometric interpretation of policy rules. It can represent all types of security policies. Policy *rules* are represented in the condition-action form, where the condition expresses a particular state which must be verified to enforce the action. A *condition*, for example, can be a specific source address of a network packet and the *action* can be that the packet is

dropped. If multiple conditions are verified in a policy at the same time, then the so-called *resolution strategy* decides which action to enforce. The resolution strategy makes its decision based on *external data*, for example, a priority that is associated with each rule. When no condition is verified, the policy applies a predefined action, the so-called *default action*. The Geometric-Model is, therefore, a 4-tuple, the rule set, the resolution strategy, the external data and the default action.

The different analysis techniques are performed on the Equivalent-Policy and are therefore independent of the network topology and the policies involved. This has the advantage that once the Equivalent-Policy is calculated all analysis types can be executed with much higher performance since the base model does not need to be instantiated every single time. Furthermore, when new types of security policies are introduced, only the computation of the Equivalent-Policy must be modified.

The Equivalent-Policy creation and the different analysis types have been implemented for performance evaluation. The implementation is designed to be reusable for future research and, therefore, exposes a well-defined API. Based on this API a graphical user interface also has been implemented. The GUI was helpful for two reasons, firstly to prove that the API is complete and secondly to be able to perform some usability tests. The performance evaluation shows that the proposed models are sufficient fast for practical usage. Furthermore, the performance is at least equivalent to the state of the art and in some aspects also faster.

This work is structure as follows: Chapter 2 presents an introduction to policies and policy types and gives detail overview of the different policy analysis techniques. Chapter 3 presents the state of the art in policy analysis. Chapter 4 presents the pros and cons of different approaches and compares them to each other. Chapter 5 presents the architecture of the proposed model and how it can be used in practice. Chapter 6 presents the mathematical model, called Geometric-Model, used to represent security policies and network topologies. Chapter 7 presents the Equivalent-Policy and how it is calculated. Chapter 8 presents the mathematical model for anomaly analysis based on the Geometric-Model. Chapter 9 presents the mathematical model for reachability analysis based on the Geometric-Model and how reachability queries are expressed and evaluated. Chapter 10 presents the mathematical model for policy comparison based on the Geometric-Model. Chapter 11 presents the implemented prototype, its components, the graphical user interface and the API. Chapter 12 presents the performance evaluation of the model and the different analysis techniques. Chapter 13 presents the conclusions and future work.

Part II

Background

Chapter 2

Policy Analysis

This chapter presents the definition of a policy and its components. It also gives an overview of the different types of policies and their main application. Furthermore, the devices that enforce security policies is introduced. The term policy analysis is defined, and the most important policy analysis technique are presented.

2.1 Policy

According to RFC3060 [2], the definition of policy is as follows: "policies are a set of rules to administer, manage, and control access to network resources." RFC3198 [1] also defines a policy as: "A definite goal, course or method of action to guide and determine present and future decisions. Policies are implemented or executed within a particular context (such as policies defined within a business unit)." Therefore, the term *policy abstraction* has been introduced to highlight different abstraction levels. A policy can be written in natural language, expressing security features of the entire network. For example the policy rule "User A can access the Internet" does not consider where user A is allocated in the network and which security controls are involved when processing this traffic. On the other hand, a policy expressed with device-specific configuration parameters is specific for one type of security control of one particular vendor. Depending on the requirements a policy is specified on this spectrum and must be elaborated accordingly.

RFC3198 [1] presents a glossary of policy-related terms, which are also used in this work and here are presented the most relevant ones. The basic building block of a policy is a *policy rule*, which is a set of *rule condition* (also named *policy condition*) and a *rule action* (also named *policy action*). A *policy condition* represents the necessary states and/or pre-requirements to perform the *rule action*. The *rule action* defines what should

be enforced by the policy. A *rule action* may result in the execution of one or more operations on the network traffic. A *policy conflict* occurs when two *rule condition* are satisfied simultaneously, and the rule actions contradict each other. A *policy anomaly* occurs when two or more *rule condition* are met simultaneously without the rule actions been necessary contradicting.

RFC3060 [2] defines different policy groups according to the purpose and intend of a policy. Various groups are Motivational, Configuration, Installation, Error and Event, Usage, Security, and Service Policies. In particular, it defines: "Security Policies deal with verifying that the client is actually who the client purports to be, permitting or denying access to resources, selecting and applying appropriate authentication mechanisms, and performing accounting and auditing of resources."

2.1.1 Type of security policy

There are many different security policies which drop, protect, alter, log, or redirect traffic. This work considers five types of security policies: *filtering*, *communication protection*, *transformation*, *logging*, and *routing*.

A filtering policy limits the access of packets to a network based on the filtering action used. There are two filtering actions, allow and deny. Allow permits a packet to enter or exit a secure network and deny blocks the access. Filtering policies can be categorized based on the expressiveness of the policy condition into: *packet filter*, *stateful* and *application-layer*. A packet filter condition is composed only of five fields: protocol type, source IP address, destination IP address, source port, and destination port. A stateful condition is also composed in addition to the five fields with a sixth field, the packet state. An application-layer condition is also composed in addition to the five fields with specific application conditions like URL, HTTP method, etc.

A communication protection policy defines the type of traffic that should be protected. The communication protection action defines which protection to apply (header and/or payload, integrity and/or confidentiality) and which cipher algorithm to use. It can also be used to create tunnels by encrypting and encapsulating packets in a new packet header. The most common communication protection policies are IPsec and SSL/TLS.

Transformation policies modify the packet header according to the transformation action. Network Address Translation (NAT) and Network Address and Port Translation (NAPT) are for example in this category. NAT modifies the source or destination IP addresses and NAPT also changes in addition to the IP addresses the port number. Another implementation of transformation policies are proxies, an HTTP proxy policies, for example, modify the HTTP header.

A logging policy is used to monitor the network traffic and create statistics about it. Network traffic can be monitored in different ways, and the logging action specifies that. The logging action can be set to count the number of packets, or save the IP addresses from where the packet comes form or where it is send to. It can also specify to save the complete packet for future usage.

A routing policy is not considered a real security policy, but it is important to consider because it alters the path packets traverse through a network. Routing policies are implemented based on a static routing tables or dynamic routing protocols. In the case of a static routing table, the routing policy specifies the concrete path through the network hop by hop. Each hop has his static routing table which specifies the interface on which a packet must be forwarded. In the case of dynamic routing protocols, the routing policy only specifies metrics based on which the routing protocol calculates the outgoing interface.

2.1.2 Security control

Network security policies are enforced by security controls, which are appliances or software modules within a computer network. Different types of security policies are enforced by various types of security controls. Filtering policies are enforced by *Packet Filter*, *Stateful Firewalls*, and *Application-layer Firewalls*. Communication protection policies are enforced by *IPsec Gateways*, *VPN Terminators*, and *SSL Client/Server*. Transformation policies are enforced by *NAT/NAPT Appliances* and *Proxies*. Logging policies are enforced by *Network Monitors*, and routing policies are enforced by *Routers*.

Security controls implement the required functionalities to enforce a security policy. They can be deployed in three different ways, dedicated appliance, software module, or virtualized. Currently, security controls are deployed for the most part as software modules in dedicated appliances. A typical firewall appliance is capable of enforcing stateful firewall policies, NAT/NAPT policies, and IPsec policies.

With the advantages in NFV, the trend goes to deploy security controls as so-called Virtual Network Functions in virtualized environments. The advantage of this approach is that the deployment is much more flexible and cost efficient.

2.2 Policy Analysis

Policy analysis is a technique which analyses policies to find errors. There are three primary policy analysis approaches: *Anomaly analysis*, *Reachability analysis*, and *Policy comparison*. Each of the approaches focuses on a different aspect of policy analysis, they

have overlapping functionalities but, in general, are complement to each other. The union of all three approaches covers the complete spectrum of policy analysis and, therefore, uncovers the grates number of errors.

Anomaly analysis searches for potential semantic errors (so-called anomalies) within a single or a set of security policies. Policy anomalies are not always errors, but can lead to unintended behaviours of the policy enforcement. They can be compared with compilation warnings of computer programs, the program complies and can usually execute, but there can be execution stages where the program fails. The analysis is based on rule relations, it searches for policy rules where the condition matches similar network packets. It can also be used for policy optimization since identified rule relations are good places to apply policy optimization techniques. Anomaly analysis has the advantage that it can be applied without any detailed knowledge of the policy. The disadvantage is that this approach has no usage without a reference policy (high-level policy or unmodified policy).

Reachability analysis searches for allowed traffic in a computer network. It evaluates if a host can communicate with a specific service or a set of services, or vice versa it evaluates if a service is reachable by which hosts. It can be performed online (by actively probing a network) and offline (by simulating an accurate representation of the network). Reachability analysis has the advantage that the network administrator can verify specific properties of a network, which at the same time is also its disadvantage. Since when a network administrator does not perform a verification where an error resides, it will never be found.

Policy comparison searches for differences between two policies, where the policies can also be written in different policy abstraction. One possible application of policy comparison is to confront a concrete configuration with the desired policy written in a high-level language. Another application is the verification of a policy change, where the modified policy is compared with the original one. The advantage of policy comparison is that it can efficiently find misconfiguration or verify the impact of a policy change. The disadvantage is that without a reference policy (high-level policy or unmodified policy) this approach has no usage.

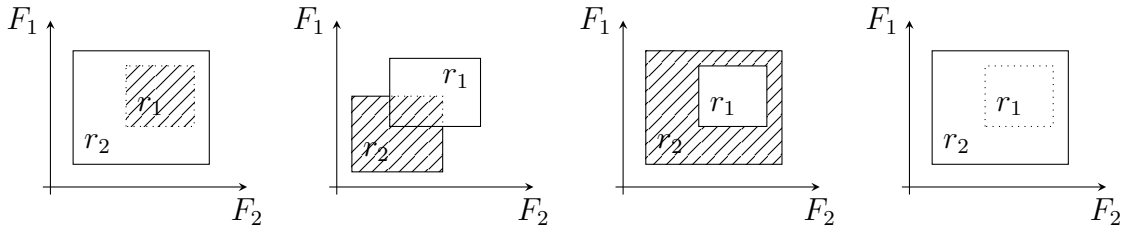
2.3 Anomaly analysis

Anomaly analysis is a technique that searches for possible anomalies within a single policy or between multiple policies. It can be applied to different types of policies, where different policy types contain different anomaly types. There is also a distinction between

anomalies from one single policy (intra-policy) and anomalies within multiple policies (Inter-policy). This section presents the three most important anomaly types, first Intra-policy, Inter-policy, and Inter-state filtering anomalies and then IPsec anomalies. The section concludes by presenting Inter-technology anomalies, which are a new anomaly class defined by this work.

2.3.1 Intra-policy filtering anomaly

Intra-policy filtering anomalies occur between rules within a filtering policy. In literature, the most important intra-policy anomaly types are shadowing, correlation, generalization, and redundancy.



(a) Shadowing (b) Correlation (c) Generalization (d) Redundancy

Figure 2.1: Graphical representation of Intra-policy anomalies.

A rule is classified as *shadowed* when there is a second rule with higher priority which matches the same packets and enforces different actions. This anomaly implies that the shadowed rule can be removed since it will never be activated. Figure 2.1a shows two rules r_1 and r_2, where r_1 is shadowed by the rule r_2. A special sub-case of the shadowing anomaly is the *general shadowing anomaly*, which occurs when a rule is shadowed by the union of multiple rules.

Two rules are classified as *correlated* when they rules have some matching packets in common and enforce different actions. This anomaly implies that one rule partly is unnecessary, and there is the possibility to simplify it. Figure 2.1b shows two correlated rules r_1 and r_2, both rules enforce a different action to where they intersect.

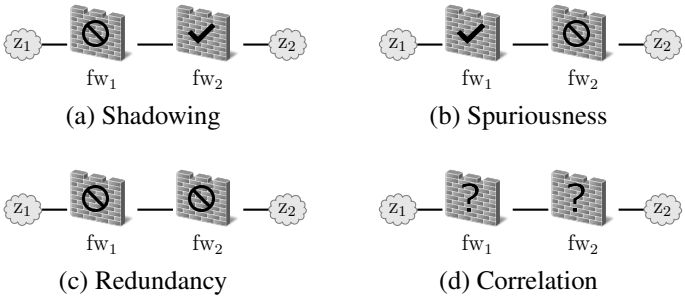
A rule is a *generalization* of a second rule when it matches the same packets as the first one and enforce different an action. Figure 2.1c shows two rules r_1 and r_2 with different actions, rule r_2 is a generalization of rule r_1.

A rule is *redundant* when it enforces the same action and matches the same packets as a second rule. This anomaly implies that the rule can be removed without changing the behaviour of the policy. Figure 2.1d shows two rules r_1 and r_2 that enforce the same

action, r_2 is redundant to r_1. A special sub-case of the redundancy anomaly is the *general redundancy anomaly* occurs when a rule is redundant to the union of multiple rules.

2.3.2 Inter-policy filtering anomaly

Inter-policy filtering anomalies occur between rules of different filtering policies. The most important Inter-policy filtering anomalies are shadowing, spuriousness, redundancy, correlation, and irrelevance.



(a) Shadowing (b) Spuriousness

(c) Redundancy (d) Correlation

Figure 2.2: Graphical representation of Inter-policy anomalies.

Two rules from two different firewalls are *shadowed* when they match the same packets and the rule from the first firewall blocks a packet that is permitted by the second rule. Figure 2.2a shows two firewalls connected in series, fw_1 has a rule which blocks the traffic and fw_2 has a rule which permits the same traffic.

Two rules from two different firewalls are *spurious* when they match the same packets and the rule from the first firewall permits the packet which is blocked by the second rule. Figure 2.2b shows two firewalls connected in series, fw_1 has a rule which permits the traffic and fw_2 has a rule which blocks the same traffic.

Two rules from two different firewalls are *redundant* when they match the same packets and both rules block the packet. Figure 2.2c shows two firewalls connected in series, fw_1 has a rule which blocks the traffic and fw_2 has a rule which also blocks the same traffic.

Two rules from two different firewalls are *correlated* when they have some matching packets in common and enforce different actions. Figure 2.2d shows two firewalls connected in series, fw_1 has a rule which permits the traffic and fw_2 has a rule which blocks the same traffic and vice versa.

A rule is classified as *irrelevant* if there is no possible traffic that can be matched by the rule. An important sub-case of the irrelevance anomaly is the *reflexivity* anomaly. A

rule is classified as reflexive when the source and destination address belongs to the same zone.

2.3.3 Inter-state filtering anomaly

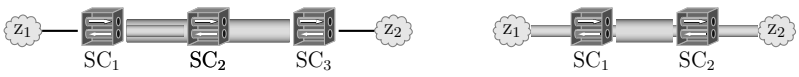
Inter-state rule anomalies are caused by stateful and stateless firewall rules. They occur when application layer protocol tries to establish a stateful connection and the connections is blocked. This anomaly can occur in two distinct moments of the connection protocol. Therefore, two Inter-state anomalies are defined.

The first type occurs when a filtering policy blocks packets during the three-way handshake. In this case, one of the three handshake packets are blocked, either the SYN, or the SYN+ACK, or the ACK packet.

The second type in this class occurs when a filtering policy blocks packets during the connection termination. In this case, there are four possible packets which can be blocked, either the first FIN, or the first ACK, or the second FIN, or the second ACK.

2.3.4 IPsec anomaly

IPsec anomalies occur between rules of IPsec policies. Therefore, they are anomalies in communication protection policies. IPsec anomalies can be divided into two categories, namely overlapping-session and multi-transform anomalies. Both anomaly categories can be found within one single policy (Intra-policy) and multiple policies (Inter-policy).



(a) Overlapping Session (b) Multi Transfrom

Figure 2.3: Graphical representation of IPsec anomalies.

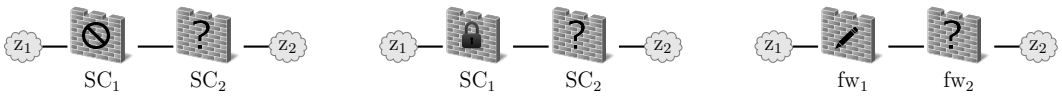
The *overlapping session anomaly* occurs when on the path from source to destination multiple IPsec sessions are established, and the earlier IPsec sessions terminate before the later ones. This anomaly causes a triangular routing, because when the outer IPsec session terminates the packet is first send back to terminate the inner IPsec session before it continues the path to its destination. Figure 2.3a shows an overlapping session anomaly where two IPsec sessions are started at security control SC_1. The inner IPsec session terminates on security control SC_2, and the outer IPsec session terminates on security control SC_3. The packet itself first is send to SC_3 where the outer IPsec header is removed

and then it is send back to SC_2 where the inner IPsec header is removed. Finally, the packet is send to its destination by traversing a second time security control SC_3.

The *multi-transform anomaly* occurs when multiple IPsec transformations are applied to a packet, and the later transformations have a weaker protection the previous one. IPsec allows to apply multiple transformations to a packet, which in some cases is desirable. For example, the ESP transformation does not protect the header and therefore afterward the AH transformation must be applied. Figure 2.3b shows an example where an AH transformation is applied after an ESP transformation. In this case, the second transformation does not improve the security properties of the packet and is therefore unnecessary.

2.3.5 Inter-technology anomalies

Inter-technology anomalies were first introduced in [22] and represent a complete new anomaly class. This new class includes three new anomalies: Blocked traffic anomaly, Encrypted traffic anomaly, and Modified traffic anomaly.



(a) Blocked traffic (b) Encrypted traffic (c) Modified traffic

Figure 2.4: Graphical representation of Inter-technology anomalies.

The *Blocked traffic anomaly* occurs when a security policy contains a rule defined for a packet that previously has been blocked by another security policy. The Blocked traffic anomaly is equivalent with the Inter-policy filtering shadowing anomaly, when both security policies are filtering ones. Figure 2.4a shows an example of this anomaly. Security control SC_1 blocks the packet for which security control SC_2 has a rule defined.

The *Encrypted traffic anomaly* occurs when a security policy contains a rule defined for a packet that previously has been encrypted by another security policy. In this case the rule can not match the correct packet because the required information is encrypted and is not accessible. Figure 2.4b shows an example of this anomaly. Security control SC_1 encrypts the packet for which security control SC_2 has a rule defined.

The *Modified traffic anomaly* occurs when a security policy contains a rule defined for a packet that previously has been modified by another security policy. Figure 2.4c shows an example of this anomaly. Security control SC_1 modifies the packet for which security control SC_2 has a rule defined.

2.4 Reachability analysis

Reachability analysis can be performed both online and offline. Online reachability analysis is carried on a deployed system by injecting test packets and verifying on different points of the network that those packets are present. Offline reachability analysis is carried on a model of the system without direct interaction with a real network.

2.4.1 Online

Online reachability is performed on a deployed system, with all security controls in place and configured. The most common applications to perform online reachability analysis are *ping* and *traceroute*. A more advanced approach are dedicated traffic generators and traffic probes installed in critical sections of the network. The traffic generators create test packets and send them to the network probes. The results of the anaysis are based on the number and types of packets revived by the probes.

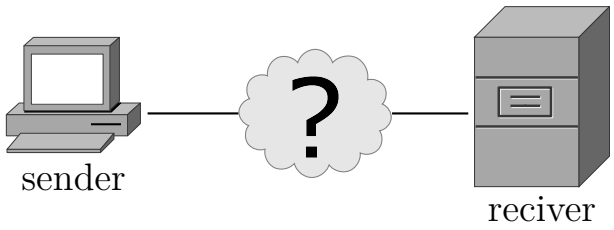


Figure 2.5: Online reachability

Online reachability analysis has the advantage that the result is correct all the time, and it is instantaneous. The correctness is guaranteed because real packets are injected into the network and are processed by all device on the path. The disadvantage is that to perform the analysis special probes have to be install within the network. Inserting these probes for the most use-cases is difficult and involves a precise planning and execution. Due to this limitation, online reachability is mainly used to verify the correct behaviour of single network components and not for policy validation of an entire network.

2.4.2 Offline

Offline reachability analysis is performed by expressing queries to the analysis system, which evaluates them and returns the answer. Reachability queries are defined in different ways varying from a query language to a graphical query definition wizard. Also the

expressiveness of different queries has a broad spectrum. A query can be very specific and request if a certain host can reach a certain service, or very general and request which hosts can reach which services. The analysis system may also include some predefined queries to verify well-known network vulnerabilities or to audit the network for certifications such as PCI-DSS.

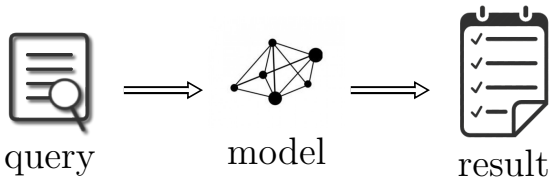


Figure 2.6: Offline reachability

Offline reachability is much more dynamic, and the planning and execution is much less invasive the the online approach. It has the advantage that it can also be applied during the design phase since the system does not need to be deployed. Additionally, different path in the network can be verified without reconfiguring the system. This gives the opportunity to test the fault-tolerance properties of the systems. The disadvantage is that the correctness of the result is based on the correctness of the model. This also implies that all parts of the network must be included in the mode. Otherwise, the analysis is limited or even impossible. Furthermore, all policies must be represented correctly by the instantiated model.

2.5 Policy comparison

Policy comparison has different application scopes, in general, it can be divided into four categories: a single policy, an entire network, Change-impact-analysis and Implementation-verification. The classification is based on two dimensions, the first dimension is the number of policies involved and the second dimension is the application type. Therefore the the different scopes can be mapped as displayed in Figure 2.7.

	Change impact (verify modifications)	Implementation verification (verify specification)
Single policy (single security control)	1	3
Multi policy (entire network)	2	4

Figure 2.7: Policy Comparison

Single policy comparison is performed between two single policies one to one. This approach is limited to one single policy type. Therefore, a filtering policy can not be confronted with a data protection policy.

Multiple policies comparison is performed between to complete network configurations and associated policies. In this case different types of policies can be involved at the same time.

Change-impact-analysis is used to verify the actual impact on a policy after inserting, removing and/or modifying a policy. Therefore, the original policy is compared with the modified one.

Implementation-verification is used to verify that a policy implementation is correct and enforces all rules as specified in the policy definition. In this scenario, the policy definition, for the most part, is written in a high-level language. In some use-cases, the policy definition is already deployed in the network by another security control. In others, the policy definition can also be a policy written by another network administrator.

2.5.1 Single policy - Change-impact-analysis

Change-impact-analysis of a single policy confronts a policy with a modified iteration of itself. This allows administrators to make modifications to a policy and verify the effective impact. Modifications to a policy include inserting a new rule, deleting a rule, modifying a rule and changing rule priority. Therefore, by applying a Change-impact-analysis to a modified policy, the network operator can verify that the modification did not have any collateral effects.

2.5.2 Multiple policies - Change-impact-analysis

Change-impact-analysis of multiple policies is used by administrators to verify the impact of a policy change within the network. It can also be used to verify changes in the network structure. In this case the complete network configuration with all involved policies is compared, and the global effect is evaluated.

2.5.3 Single policy - Implementation-verification

Single policy Implementation-verification is used to verify that a policy implementation is correct respect to a policy definition. The policy definition can be expressed in a high-level language which captures the general concepts, or in a low-level language defined by another network administrator or used by another security control. The policy implementation is then evaluated, and all violations of the policy definition are highlighted.

2.5.4 Multiple policies - Implementation-verification

Multiple policies Implementation-verification is used to verify that an entire network implementation is correct respect to a network policy definition. Also, in this case, policy definition is expressed in a high-level language which captures the general concepts of the entire network. The complete network configuration with all involved policies is evaluated, and all violations of the policy definition are highlighted.

Chapter 3

State of the art

This chapter presents the state of the art in policy analysis. First the research regarding anomaly analysis is presented, starting from the analysis of filtering policies to data-protection policies. Afterwards, the research regarding reachability analysis is presented. Than the different papers on policy comparison are presented.

3.1 Anomaly analysis

The greatest research effort in anomaly analysis is spent on single filtering policies and computer networks containing only filtering policies. There are only a few research approaches that incorporate different types of security policies into a single analysis model. Research efforts apart of filtering policies have been concentrated on analysing IPsec configurations. Some of these works also include filtering policies, but they are the minority.

This section is therefore divided into two parts, first the state of the art of filtering policy anomaly analysis is presented and afterwards the research on IPsec configuration analysis.

3.1.1 Filtering

Qian et al. [23] had one of the first attends to resolve policy misconfigurations by proposing a framework for automatic access control list (ACL) analysis. The propose algorithm can detect and remove redundant rules and discover and repair inconsistent rules. Furthermore, it can also merge overlapping or adjacent rules, and rewrite ACLs to be more readable. The shortcoming of this paper is that it does not consider all types of policy anomalies introduced in future works.

Al-Shaer and Hamed [24] first introduce the anomaly analysis of filtering policy. They defined five filtering rule relations: completely disjoint, exactly matching, inclusively matching, partially disjoint and correlated. Based on these rule relations they presented the definitions for Intra-policy rule anomalies (shadowing, correlation, generalization and redundancy). The shadowing anomaly occurs when two rules match the same packets but enforce different actions. The correlation anomaly occurs when two rules match similar packets but enforce different actions. The generalization anomaly occurs when a second rule matches the same packets as the first one but not vice versa, and they enforce different actions. The redundancy anomaly occurs when two rules match the same packets and enforce the same action.

Al-Shaer et al. also propose the definition for anomalies between serially-connected packet filters [25][26] and named it inter-policy anomalies. Inter-policy anomalies as defined by Al-Shaer include shadowing, spuriousness, redundancy, correlation and irrelevance. The shadowing anomaly occurs when two rules from two different packet filters match the same packets and the first rule blocks a packet that is permitted by the second rule. The spurious anomaly occurs when two rules from two different packet filters match the same packets and the first rule permits a packet that is blocked by the second rule. The redundancy anomaly occurs when two rules from two different packet filters match the same packets and both rules block the packet. The correlation anomaly occurs when two rules from two different packet filters match similar packets and enforce different actions. The irrelevant anomaly occurs when a rule is defined for which exist no possible packet.

The anomaly classification of Al-Shaer et al. has been the base of alternatives models and classification schemas. These research has shown that the results of Al-Shaer are correct and can be applied to real-world applications. However, all of this research focuses only on packet filters and ignore other security policies. Firecrocodile [27], propose by Lehmann et al., analyses PIX firewall configurations for Intra-policy anomalies. Additionally, it also verifies the configuration for policy violations. Yuan et al. propose FIREMAN [28], an Intra-Policy analysis tool based on binary decision diagrams (BDDs) to represent packet filtering policies. The tool is also able to verify that an end-to-end policy is correctly implemented. Jeffrey et al. [29] propose to reduce the complexity of the problem by using an SAT-solver instead of BDDs. To proof, their claim they reimplemented FIREMAN with their SAT-solver. Hu et al. [30], [31] propose a tool called Firewall Anomaly Management Environment (FAME). It uses a rule-based segmentation technique and a grid-based representation to identify policy anomalies. Krombi et al. [32] propose an automata-based approach to develop three analysis procedures. Their algorithm can verify completeness, detect anomalies and functional discrepancies between several implementations of a security policy.

There are also research proposals which focus on a formal proof of their analysis model. Capretta et al. [33] present one of the first formal anomaly detection algorithms for packet filter policies. The disadvantage of the propose algorithm is that it can not distinguish between different anomaly types. Their implementation has been verified with the Coq Proof Assistant and is sound and complete. Abbes et al. [34] suggest using an inference system for detecting Intra-policy anomalies. The inference system constructs a tree representation of the policy and stops the construction of a particular branch when no anomaly can be found. The resulting classification tree contains potential rule anomalies in its leaves. Basumatary et al. [35] represent firewall rules by a topological-temporal model and use a model checker to verify the firewall policy.

Other researchers inspired by these results focused their efforts to find new types of Intra- and Inter-policy anomalies. Golnabi et al. [36] propose two new non-systematic Inter-policy anomalies: *Blocking existing service* and *Allowing traffic to non-existing service*. The first anomaly occurs when legitimate traffic from a trusted network to a valid service is blocked. The second anomaly occurs when traffic to a non-existing service is permitted. Basile et al. [37] propose two new anomaly types: *general redundancy* and *general shadowing*. The general redundancy anomaly occurs when a rule is redundant to the union of a set of rules. The general shadowing anomaly occurs when a rule is shadowed by the union of a set of rules.

An important aspect that was neglected by most research efforts is the introduction of new security policy types in the analysis process. Garcia-Alfaro et al. [38] propose to integrate network intrusion detection systems (NIDS) into anomaly analysis tools. The propose model is based on their previous work [39] and has been integrated into the MIRAGE [40] tool. Their model can detect both Intra- and Inter-policy packet filter rule anomalies. Additionally, it can also verify, based on its source and destination address, which security controls are on the path of a given packet. A solution for Intra-policy analysis of stateful firewalls has been propose by Cuppens and Garcia-Alfaro [41], [42]. They also introduce two new anomaly classes, namely Intra-state and Inter-state rule anomalies. The Intra-state anomalies include the stateless anomalies as presented by Al-Shaer and two new anomaly types for stateful rules. The first stateful anomaly occurs when a security control blocks a packet during the three-way handshake. The second stateful anomaly occurs when a security control blocks a packet during the connection termination. The Inter-state anomalies occur when a stateless rule blocks packets required by a stateful connection. An example of an Inter-state anomaly is for example when the FTP protocol operates in active mode and the data channel traffic is blocked. Basile et al. [43] present an algorithm for anomaly analysis of application-level firewall policies. Their algorithm can identify all anomaly types defined by Al-Shaer for packet filter policies,

but applied to application-level firewall policies. The main difference between packet filter policies and application-level policies is the possibility to define regular expressions within the rule conditions. Therefore, the main contribution of this paper is the calculation of rule relations based on regular expressions. The presented algorithm transforms regular expressions into deterministic automata, which can be manipulated much more efficiently.

Other research approaches also tried to resolve the detected anomalies. Liu et al. [44] focus in their paper only on redundant rules in packet filter configurations. Their algorithm can identify and remove two types of redundant rules: upward redundant rules and downward redundant rules. Rules which are never matched are upward redundant rules, this definition is similar to the definition of shadowed rules by Al-Ahaer. Rules which are matched but enforce the same action as rules with lower priority are downward redundant rules. Cuppens at al. [45] also propose an algorithm that is capable of resolving Intra-policy anomalies. Like Liu et al. they also only handle shadowed and redundant anomalies. The authors extend their model to support also Inter-policy anomalies [46]. Additionally, the authors introduce the reflexivity anomaly, which is sub-case of the irrelevance anomaly. It occurs when the source and destination address are part of the same filtering zone. Abedin et al. [47] generate a new anomaly free policy by reordering and splitting conflicting rules. The presented algorithm reduced the rule number and, therefore, increase the efficiency of the firewall. Ferraresi et al. [48] propose to resolve all policy conflicts by rewriting the policy with only disjoint rules. Their algorithm does not operate completely autonomous. The user is required to decide which rule to delete or modify when the algorithm can not take a decision. Prometheus is an open-source firewall analysis tool propose by Oliveira et al. [49]. The tool can perform anomaly and reachability analysis of firewall policies. The algorithm can detect redundancy, shadowing, generalization and correlation anomalies. Furthermore, a verification is performed to ensure that all possible paths in the network enforce the same security policies. The anomaly resolution is not performed autonomous instead possible solutions are suggested, and the user can select which to apply. Saadaoui et al. [50] propose a formal approach for detecting and resolving firewall misconfigurations. Their approach also detects anomalies between multiple rules and not only rule pairs. Furthermore, they introduce a new classification of firewall anomalies: superfluous rule-class anomalies and conflicting rules-class anomalies. Superfluous rule-class anomalies include the shadowing and redundant anomaly and conflicting rules-class anomalies include the correlation and generalization anomaly. The new classification is used because superfluous rule-class anomalies can be resolved by their algorithm.

3.1.2 Data-Protection

IPsec policy anomaly detection was first introduced by Fu et al. [51]. The authors propose to identify anomalies by verifying that the policy implementation satisfies the requirements of a desired policy. The policy implementation and the desired policy are written in a high-level language. Additionally, to the anomaly identification they also propose an anomaly resolution process. The anomaly resolution process creates a new policy based on the policy implementation that satisfies the desired policy.

Al-Shaer [52] proposes an anomaly classification scheme based on the work of Fu et al. [51]. He creates a complete taxonomy [53] of rule anomalies for packet-filter and IPsec policies. This research is the only one that combines anomaly analysis for two different security controls. The taxonomy also includes two new IPsec anomalies: overlapping-session and multi-transform anomaly. The overlapping-session anomaly occurs when from the same source two or more nested IPsec sessions are established to different destinations and the inner session terminates on a nearer host then the outer one. The multi-transform anomaly occurs when an encapsulated IPsec traffic is encrypted a second time and the second protection is weaker than the first one. Additionally to the classification, Al-Shaer also propose a new model for detection them. The new model can detect packet filter and IPsec anomalies, and is based on OBDD (Ordered Binary Decision Diagrams).

Another anomaly classification model for IPsec policies has been presented by Li et al. [54]. This model is designed to handle similar Intra- and Inter-policy anomalies as defined by Al-Shaer. However, the two models are not compatible to each other. Li et al. use a slightly different classification that has a clearer structure and is, therefore, easier to implement.

An improved version of Al-Shaer's [52] IPsec anomaly classification has been presented by Niksefat and Sabaei [55]. Their detection algorithm identifies only IPsec anomalies and uses Binary Decision Diagrams (BDD) to represent the policies. The main advantage of this approach over Al-Shaer's is that it has better performance. Furthermore, Niksefat and Sabaei also include an anomaly resolution algorithm in their solution.

Valenza et al. [56] present a novel classification of communication protection policy anomaly. The propose model allows the detection of some anomalies arising from the interactions between various protection protocols that work at different layers of the ISO/OSI stack (IPSec, TLS, SSH and WS-Security), security proprieties and communication scenario such as end-to-end connection, VPNs and remote access.

3.2 Reachability analysis

Reachability analysis can be performed online or offline. Online reachability analysis, in general, is performed by injecting test packets into a deployed network and verifying that they arrive at their destination. Offline reachbility analysis is performed by instantiating and querying a model of the network. Research efforts are mainly focused on offline reachability analysis techniques.

Although online reachability analysis has been performed in practice for a long time with tools such as *ping*, *traceroute*, and *tcpdump*, the scientific community has given only a little attention to this topic. In recent years, there has been an effort to create new commercially available tools for online reachability analysis. The most important are SATAN [57], Nessus[58], and FTester[59]. These solutions are based on a traffic generator and a traffic analyser. The traffic generator sends packets to the traffic analyser that creates a statistic on which packets have arrived and which not. In literature solutions for online reachability analysis are presented by El-Atawy et al. [60] and Al-Shaer et al. [61]. Their traffic generator first analyses the security policy and then generates the most relevant packets. The limitation of this two solutions is that they work only with single packet filters. Brugger et al. [62] propose a tool for model-based conformance testing of firewalls called HOL-TESTGEN/FW. Their approach uses a test-policy from which the effective reachability queries are generated. Test-policy have the advantage that queries can be written in a more abstract level, and the reachability analysis is closer to a policy comparison.

One of the first Offline reachability solutions was propose by Mayer, Wool, and Ziskind [63]. This firewall analysis engine was called FANG (Firewall Analysis Engine) and is designed for computer networks containing only packet filters. It takes as input the network topology and the packet filter configuration files. Reachability queries are defined by a user interface, and FANG evaluates them. The query interface has been improved in the extended versions of the paper [64] and [65]. The newer solutions also generated the most relevant queries automatically.

Eronen et al. [66] propose to use logic programming and a generic inference engine to analyse Cisco ACLs. Their approach has the advantage that reachability algorithms are not hard-coded. New rule analysing functions can easily be added, and the expert knowledge can be expressed in a compact form. Furthermore, their model can also analyse the rule structure, for example if a rule is never matched.

Hazelhurst [67] presents a model for reachability analysis and policy comparison. The model represents packet filter rules internally as BDDs and has a GUI for human interaction. To perform a reachability analysis, the user expresses a query as a Boolean

expression, and the result is displayed in tabular form. The user can express queries for packets which are allowed and denied. Furthermore, the output can be formatted to achieve different levels of abstraction.

Xie et al. propose a reachability analysis algorithm based on graph theory and dynamic programming [68]. The algorithm supports static NAT, routing, and filtering and operates in one of two modes: upper and lower bound. The upper bound verifies that at least one possible path ensures reachability and the lower bound verifies that all possible paths ensure reachability. However, it is only a theoretical model and has no practical validation. To overcome this limitation Bandhakavi et al. [69] present an extension of this work. The extended algorithm is based on a general model for representing firewalls, packet filtering and transformation rules. It also handles policies defined for the source addresses and filtering states.

Marmorstein and Kearns [70] [71] [72] propose a tool for analysing IPtables policies named ITVal. The main contribution of this research is to be the first open source implementation of a firewall analyser capable of performing simple reachability queries. The tool represents IPtables policies as multi-way decision diagrams (MDD) where each level represents one attribute of the rule condition. Since the tool was designed for IPtables policies, it is also able to handle state-full conditions. Unfortunately, the propose algorithm is not able to perform an Inter-policy analysis and can not handle transformation policies.

Matousek et. al. [73] present a reachability analysis model for networks containing packet filters and dynamic routing policies. The authors propose to represent ACLs as Interval Decision Diagrams (IDDs) [74]. IDDs are an efficient data structure on which different operations can be performed. Routing information is modelled as additional rules in the IDD and is used to test link failures.

Khakpour and Liu [75] propose Quarnet, a reachability analysis tool. Their solution calculates the network reachability metrics and supports connectionless (stateless router/-firewall and static NAPT) and connection-oriented transport protocols (stateful router/-firewall and dynamic NAPT). The network reachability metrics are based on Firewall Decision Diagram (FDD) and used to solve reachability queries. The computation of the FDD takes a long time but afterwards the reachability queries are resolved very fast.

Prometheus is an open-source firewall analysis tool propose by Oliveira et al. [49]. The tool can perform anomaly analysis but also performs a flow connectivity analysis. The algorithm verifies that a server is accessible from different locations in the network. First all possible routes between the nodes are calculated and eliminates all routes on which a firewall blocks the traffic. If no route remains, Prometheus points out which firewall rules block the traffic.

Nelson et al. [76] propose a reachability analysis tool named Margrave. The tool can perform basic reachability queries of on firewall policies and can also be used to perform a Change-impact-analysis. The authors define a first-order language to represent both policies and reachability queries. Their model is also able to handle NAT and routing policies.

Mai et al. [77] present a tool for reachability verification called Anteater. Their algorithm collects data plane information from network devices and models data plane behaviour as instances of satisfiability problems. Reachability queries are represented as a boolean formula which is evaluated by an SAT solver. The reachability query is true if and only if the Boolean formula is satisfied.

Al-Shaer et al. [78] propose an algorithm for general reachability verification. The authors have implemented the algorithm in a tool and called it ConfigChecker. The tool models the global end-to-end behaviour of a network and performs reachability verification. Furthermore, it can also identify bogus entries, verify IPsec tunnel integrity and discover backdoors.

Sveda et al. [79] propose another theoretical approach for resolving network-wide reachability queries. Their research is not based on an ad-hoc algorithm like [68] and [69], but use traditional graph-based algorithms such as Floyd-Marshall. They encode the reachability problem for an SAT instance and solve it with automatized solvers. Their solution supports routing and filtering policies but is not able to handle transformation policies.

Kazemian et al. [80] propose an extension of Xie's work [68]. Their solution is based on the so-called "Header space" information of packets. It supports filtering, routing and transformation policies, but only at the lower layer of the ISO/OSI stack.

3.3 Policy comparison

In literature exist two distinct approaches that utilize policy comparison: Implementation-verification and Change-impact-analysis. Implementation-verification compares a deployed policy with the desired one and verifies that the desired policy is correctly implemented. Change-impact-analysis evaluates the impact of a modification to a security policy. Research efforts have been focused first on Implementation-verification and only later on Change-impact-analysis. Implementation-verification has been applied to both filtering and IPsec policies, wears Change-impact-analysis is applied only to filtering policies.

Guttman [81] presents one of the first tools for Implementation-verification. The author propose a simple language for expressing a global filtering policy. The propose model compares the global policy with a network topology containing access control lists and verifies that it is implemented correctly. The verification algorithm compares each policy rule with all non-cyclic path in the network. This model has also been extended [82] to support IPsec gateways within the network.

Fu et al. [51] present the first solution to verify the correct implementation of IPsec policies. Their algorithm compares a high-level security policy, describing an implementation, with a desired end-to-end policy. However, it is designed only for implementation validation and not for Change-impact-analysis. In general is this approach more directed towards anomaly analysis then policy comparison.

Hazelhurst [67] presents a model for reachability analysis and policy comparison. The model represents packet filter rules internally as BDDs and has a GUI for human interaction. The policy comparison algorithm can compare an original policy with a modified one. The result of shows if the two policies are equivalent or contain differences. The differences can be displayed in two modes, *newdeny* and *newallow*. The newdeny mode highlights the packets which are blocked by the modified policy, but where allowed by the original one, the newallow mode does the opposite.

Liu et al. [83] present a firewall verification tool that verifies a given property within a packet filter policy. Their algorithm is designed for offline firewall debugging and troubleshooting. It first computes an FDD representation of the policy and afterwards the verification is performed on the FDD. The verification compares the action saved in the leafs of the FDD with the desired action given by the property. The authors have implemented the algorithm and tested it for performance. Although it has excellent performance, it is limited to one single firewall and cannot verify the correct implementation of a complete network.

Krombi et al. [32] propose to represent a packet filter policy as an automaton. Their algorithm can verify completeness, detect anomalies and functional discrepancies between several implementations of a security policy. The comparison of two packet filter policies is based on three fundamental steps. First both policies are transformed into automata. Then they are combined into a single one, where each leaf notates the actions of both automatons. The final step is to verify that all leafs contain the same actions and each leaf where the actions contradict each other a discrepancy between the input policies is present.

Liu et al. [84] and [85] propose an algorithm that compares filtering policies written by two different network administrators. First the two policies are converted into FDDs and afterwards the two FDDs are compared to each other. The comparison algorithm

confronts the actions in the leafs of the two FDDs and highlights each leaf where the action differs. The differences found in the two policies are presented to the administrators, which must choose the correct action to enforce. This approach is simmilar to the Change-impact-analysis because one policy can be seen as the original and the other one as the modified policy.

Yin and Bhuvaneswaran [86] are the first ones to evaluate the impact of rule changes on the policy. Correlations between rules are modelled as spatial relations, and filtering policies are represented by the so-called SIERRA tree. A SIERRA tree is similar to an FDD, each level of the tree represents a dimension of the special division. The algorithm can only verify single modifications to the policy. The verified modifications include adding a new rule and removing or modifying an existing rule. The performance of the algorithm is not very promising because the verification of a policy with only 30 rules takes already several seconds.

The Margrave Tool by Nelson et al. [76] can perform Change-impact-analysis of single firewall policies. The tool represents firewall policies with a first-order language and is also able to perform reachability analysis. The implemented algorithm compares a modified firewall policy with the original one and lists all packets which are treated differently by the two policies. Although the algorithm can perform reachability queries on entire networks with multiple firewalls and NAT, the Change-impact-analysis is designed only for single policies.

Liu et al. have propose an algorithm for Change-impact-analysis of firewall policies [87] [88]. The algorithm first converts the packet filter policies into FDDs and eliminates all overlapping rules. It supports four basic modification operations: rule deletion, rule insertion, rule modification, and rule swap. The result of the algorithm is an accurate representation of the effective modification. The algorithm is also able to correlate the impact with a high-level security policy. However, a representative performance evaluation of the algorithm has not been presented.

Youssef et al. [89] propose to use a inference system for automatic Implementation-verification. The propose algorithm verifies that a filtering policy is sound and complete compared to the desired policy. All differences are presented to the user that must resolve them manually. Youssef et al. [90] propose an improved algorithm to check also statefull firewall policies.

Chapter 4

State of the art - summary

This chapter presents the advantages and disadvantages of the different research approaches in the state of the art. First advantages and disadvantages of the research approaches regarding anomaly analysis are presented. Afterwards, advantages and disadvantages of the research regarding reachability analysis are presented. Then the advantages and disadvantages of different papers on policy comparison are presented. Finally, a summary of all the articles is made by confronting the similarities and uniqueness.

4.1 Anomaly analysis

Anomaly analysis is one of the first approaches to identify security policy errors and, therefore, includes also the most research papers. Another important consideration is that initially anomaly analysis focused mainly on single filtering policies. The analysis of entire computer networks containing filtering policies has gained less attention. Research efforts have also been concentrated on analysing IPsec configurations. However, there are only a few research approaches that incorporate different types of security policies into a single analysis model. Some of these models also include filtering policies and IPsec configurations, but they are the minority. This section is therefore divided into two parts. First advantages and disadvantages of the state of the art on filtering policy anomaly analysis are presented. Afterwards the advantages and disadvantages of research on IPsec configuration analysis.

4.1.1 Filtering summary

For the last decade, there have been many publications regarding filtering policy analysis. In general, can be concluded that filtering policy analysis is a well-studied field. It is mature enough to apply the research outcomes to solve real policy problems. However, some minor details could be improved, and some functionalities can be added.

Table 4.1 gives an overview of the main properties of these articles, and this subsection summarizes the overall research outcome. In the first column-group, the supported security policy types are marked. The second column-group marks if the presented analysis model can also be applied for Inter-policy analysis and if it can resolve the identified anomalies. The third column-group represents how the analysis model has been tested and evaluated.

The main conclusion from this summary is that for the biggest part only packet filter policies have been considered. Only a small part of the publications also considered other filtering policy types, such as stateful and application layer firewalls. This is not a real disadvantage since these publications have a an excellent coverage of the analysed policies. Therefore, from a general perspective, all relevant filtering policies are present. However, the fact that application layer firewalls have been ignored in the most recent publications may be an indication that the research focus points in the wrong direction.

A more important conclusion is that not even half of the articles considered inter-policy anomalies. This is a major disadvantage since without considering the interactions of a policy with other policies the performed analysis is incomplete. Inter-policy anomalies have a significant effect on the global network behaviour and must therefore always be considered.

Furthermore, although there are research results regarding anomaly resolution, they focus only on Intra-policy anomalies and are only able to resolve redundant rules. Therefore can this topic not been considered mature enough and more research effort should be focused on it.

Finally, the performance evaluation of the presented algorithm also leaks consistency. Although almost all analysis algorithms have been implemented and tested, the test cases are not well documented. Therefore, a conclusive performance comparison is impossible. Some papers also have performed an empirical and theoretical evaluation of the presented algorithm.

Paper	Packet Filter	Stateful FW	Application FW	Inter-Policy	Resolution	Prototype	Performance tests	Empirical eval.	Theoretical eval.
[24]	✔								
[25]	✔			✔		✔	✔	✔	
[26]	✔			✔		✔	✔	✔	
[27]	✔					✔	✔	✔	
[28]	✔			✔		✔	✔		
[29]	✔					✔	✔		
[30]	✔				✔	✔	✔	✔	
[31]	✔				✔	✔	✔	✔	
[32]	✔								
[33]	✔					✔	✔		✔
[34]	✔					✔	✔		
[35]	✔					✔	✔		
[36]	✔					✔	✔	✔	
[37]	✔					✔	✔		
[38]	✔			✔		✔	✔		
[39]	✔			✔		✔	✔		
[40]	✔			✔		✔	✔		
[41]	✔	✔				✔	✔	✔	
[42]	✔	✔				✔	✔	✔	
[43]	✔		✔		✔	✔	✔		
[44]	✔				✔				
[45]	✔				✔				✔
[46]	✔				✔	✔	✔		✔
[47]	✔				✔				✔
[48]	✔				✔	✔	✔		✔
[49]	✔			✔	✔	✔			✔
[50]	✔					✔	✔		

Table 4.1: Filtering summary

4.1.2 Data-protection summary

The research of policy analysis techniques for data-protection policy has got less attention then filtering policies. The different papers only focus on one data-protection policy type, namely IPsec policies. There was only one publication which also considered Inter-protocol data-protection anomalies and, therefore, the interaction with other policy types.

Paper	IPsec	Filtering	Inter-protocol	Resolution	Prototype	Performance tests	Empirical Eval.	Theoretical Eval.
[51]	✔			✔	✔			
[52]	✔	✔			✔	✔	✔	
[53]	✔	✔			✔		✔	
[54]	✔	✔			✔			
[55]	✔			✔	✔	✔		
[56]	✔		✔	✔	✔	✔		

Table 4.2: Data-protection summary

Table 4.2 gives an overview of the key-properties of those research approaches. In the first column-group, the supported security policy types (IPsec and filtering) are marked. The second column-group marks if the presented analysis model can also be applied for Inter-protocol data-protection analysis and if it can resolve the identified anomalies. The third column-group represents how the analysis model has been tested and evaluated.

About half of the presented algorithms is also capable of handling other policy types beside data-protection. They also included filtering policies and can, therefore, analyse computer networks which include data-protection and filtering policies at the same time. This is a significant advancement but still not enough, because computer networks with transformation policies can invalidate the whole analysis result. Interestingly researchers that considered filtering policies did not propose any anomaly resolution and vice versa.

All proposed algorithms have been implemented and about the half performed a performance evaluation. However like in the case of filtering policies, also these papers do not explain well enough how the performance evaluation was conducted and how the test policies have been created. Therefore, is it impossible to compare the results to each other and decide which algorithm performs the best.

4.2 Reachability analysis

Reachability analysis can be performed online or offline. Online reachability analysis, in general, is performed by injecting test packets into a deployed network and verifying that they arrive at their destination. Offline reachability analysis is performed by instantiating and querying a model of the network. Research efforts are mainly focused on offline reachability analysis techniques.

Papers	Packet Filter	NAT/NAPT	Routing	On-line	Off-line	Prototype	Performance tests	Empirical Eval.	Theoretical Eval.
[60]	✔			✔		✔	✔		
[61]	✔			✔		✔	✔		
[62]	✔	✔	✔	✔		✔	✔		
[63]	✔				✔	✔			
[64]	✔				✔	✔		✔	
[65]	✔				✔	✔		✔	
[66]	✔				✔	✔			
[67]	✔				✔	✔			
[68]	✔	✔	✔		✔				
[69]	✔	✔	✔		✔				
[70]	✔				✔	✔			
[71]	✔				✔	✔			
[72]	✔				✔	✔			
[73]	✔		✔		✔	✔			
[75]	✔	✔	✔		✔	✔	✔	✔	
[49]	✔				✔				
[76]	✔	✔	✔		✔				
[77]	✔	✔	✔		✔	✔	✔	✔	
[78]	✔				✔				
[79]	✔		✔		✔				
[80]	✔				✔	✔	✔	✔	

Table 4.3: Reachability analysis summary

Table 4.3 presents the key properties of the presented papers. In the first column-group, the supported security policy types are marked. The second column-group marks

if the presented reachability analysis is performed online or offline. The third column-group represents how the analysis model has been tested and evaluated.

This summary also includes the papers on online reachability analysis. However, those are not relevant for the proposed solution. Research on reachability analysis which does not consider security aspects of computer networks were not considered, therefore all included papers can manage filtering policies.

Nearly half of the presented reachability algorithms can also handle NAT/NAPT and routing policies. However, none of them can manage data-protection policies or application layer policies. Therefore computer networks with such security policies can not be analysed. This is a significant disadvantage since data-protection policies and application layer policies are fairly common in today computer networks.

Although the majority of the algorithms have been implemented, only a few have conducted a performance analysis. Furthermore, the results are not very conclusive, especially because like in other research papers on policy analysis the methodology was not presented clearly. Therefore, also, in this case, it is not possible to compare the performance of the different algorithms.

4.3 Policy comparison

In literature exist two distinct approaches that utilize policy comparison: Implementation-verification and Change-impact-analysis. Implementation-verification compares a deployed policy with the desired one and verifies that the desired policy is correctly implemented. Change-impact-analysis evaluates the outcomes of a modification on a security policy.

Research efforts have been focused first on Implementation-verification and, only later, on Change-impact-analysis. Implementation-verification has been applied to both filtering and IPsec policies, wears Change-impact-analysis is applied only to filtering policies. It is important to point out that all presented papers on Change-impact-analysis focus only on single policy analysis. There has been no solution that supports multiple policies analysis. Therefore Change-impact-analysis can not be applied to entire computer networks.

Table 4.4 presents the key properties of the presented papers on policy comparison. In the first column-group the supported security policy types are marked. The second column-group marks if the presented policy comparison model is designed for Change-impact-analysis and/or Implementation-verification. The third column-group represents how the analysis model has been tested and evaluated.

Papers	Packet Filter	Stateful FW	IPsec	Change Impact Ver.	Implementation Ver.	Prototype	Performance tests	Empirical Eval.	Theoretical Eval.
[81]	✔				✔	✔	✔	✔	
[82]	✔		✔		✔	✔			
[51]			✔		✔	✔			
[67]	✔				✔	✔			
[83]	✔				✔	✔	✔	✔	
[90]		✔			✔	✔			
[32]	✔				✔				
[84]	✔			✔		✔	✔	✔	
[85]	✔			✔		✔	✔	✔	
[86]	✔			✔		✔	✔		
[76]	✔			✔		✔			
[87]	✔			✔					
[88]	✔			✔		✔	✔	✔	
[89]	✔			✔		✔	✔		

Table 4.4: Policy comparison summary

Like in the other field of policy analysis the majority of work focuses on filtering policies. Only a few of them also consider stateful or IPsec policies, and none of them included application layer policies. Therefore also, in the case of policy comparison, there is no solution that can be applied to all computer networks.

The application of the policy comparison algorithm is divided 50-50 between *Change-impact-analysis* and *Implementation-verification*. In recent years, there is no clear indication that one of the two application has gained more attention than the other. Interestingly in all papers only one of the two comparison scope has been chosen. No research approach tried to propose a comparison model that can perform both scopes.

Nearly all of the presented algorithms have been implemented, and the majority has also been validated for performance. However, like for all other policy analysis techniques, it is not possible to confront the performance of the different algorithms to each other.

4.4 Summary

After heaving presented the different research papers and compared them to each other, based on their analysis technique, it is time to perform a global comparison and summarize the state of the art based on different indicators. This section compares the state of the are in policy analysis based on interoperability, performance evaluation, user interface, and implementation.

4.4.1 Interoperability

The comparison of the different research approaches clearly shows that there is almost no interoperability between the different analysis techniques. For the most part the presented analysis model focus on only one analysis technique. Only a few researchers tried to incorporate more than one policy analysis technique into their model. This has a significant disadvantage for future research and practical usage.

The negative impact on future research is present because improvements in one analysis technique can not be directly applied to another one. For example, if a new security policy category is introduced it must be modelled for each of the three analysis techniques. This can be a significant overhead and substantial research time is lost.

Form a practical perspective this situation is suboptimal since to cover the complete analysis spectrum multiple tools must be instantiated. Since different analysis techniques are designed to identify specific policy errors, all three techniques are required. Therefore, the execution time increases. Furthermore, supporting software modules, such as policy parsers and graphical interfaces, must be implemented multiple times.

4.4.2 Performance evaluation

The performance evaluation is performed by only a few papers and for the most part, the results are not reproducible. Furthermore, it is not clear how the test policies have been generated and what is the topology of the network. The description of the test cases is very superficial or not present at all.

The main problem for a representative performance comparison between the different algorithms is that there is no clear description of how to perform a representative performance evaluation. A unified analysis model may not solve this problem completely but it provides a common base that enables a better comparison and evaluation.

4.4.3 User interface

Although the user interface is not the main consideration to evaluate research efforts, it should not be ignored. Only a few research papers include a description of their user interface and how end-users must interpret the analysis results. This is a crucial aspect because when the user is not able to read and understand the analysis results, they are useless.

The user interface is necessary for the end user in two distinct ways. Firstly, to define the network topology and the security policies. Should a user not be able to model the input data in a efficient way he is not able to perform an analyse. Secondly, to represent the analysis results in a clear way. The most advanced analysis model has no sense when the analysis result can not be represented in an clear way.

4.4.4 Implementation

For a complete validation of a policy analysis algorithm, its implementation is necessary. Unfortunately, not all algorithms have been implemented. Therefore, a correct validation is impossible. Another important consideration is if the implementation is available to the public and other researchers. Here the situation is even worse because only a handful off algorithms have been released, either as commercial products or as open source.

A widely available implementation of the research outcomes has two major advantages. Firstly, it gives application developers easy access to new and correct functionalities for their products. Secondly, it guarantees to support future research efforts in this field, since researchers have a working implementation which they can work on and improve. For this reason, an open-source implementation is required for an effective research validation and to reduce future research efforts.

Part III

Proposal

Chapter 5

Requirements, Design and Contribution

This section introduces the proposed solution and gives a brief overview of the function blocks and their key features. First a general description of how the solution can be used in practice is given, different workflows and use-cases are presented. Then the requirements and main features of the solution are shown. Afterwards, the single function blocks of the solution are presented and described how they interact with each other. Finally, the contributions to the state of the are presented.

5.1 Policy Analysis workflows

A security policy goes through different stages, including the design stage, the implementation stage, the test stage and the maintenance stage. In the design stage, a security policy is designed and defined. In the implementation stage, the policy is implemented in a security control. In the test stage, the enforced policy is tested and validated. In the maintenance stage, the security policy is maintained to ensure continuous and correct operation.

Each stage can introduce new policy errors and requires different analysis techniques to find and resolve them. The various policy analysis techniques are designed for different purposes and can help at different stages. Anomaly analysis searches for potential errors within a policy, it is based on rule relations. Reachability analysis verifies that specific properties of a security policy are correct, it is based on reachability queries. Policy comparison identifies the differences between two policies or between entire network configurations. During a complete life-cycle of a security policy, all three analysis approaches are applied.

This section presents four analysis workflows namely policy generation, policy verification, policy modification and policy troubleshooting. It also shows that all three analysis techniques are required when performing these workflows. The policy generation workflow is applied when an entire network is designed, and new policies are created. The policy verification workflow is used to verify the correctness of a network deployment and its security policies. The policy modification workflow is applied when a policy or network modification must be validated. The policy troubleshooting workflow is used to find the cause of a network problem. Figure 5.1 shows the different workflows, at which stage they are applied and how they are correlated.

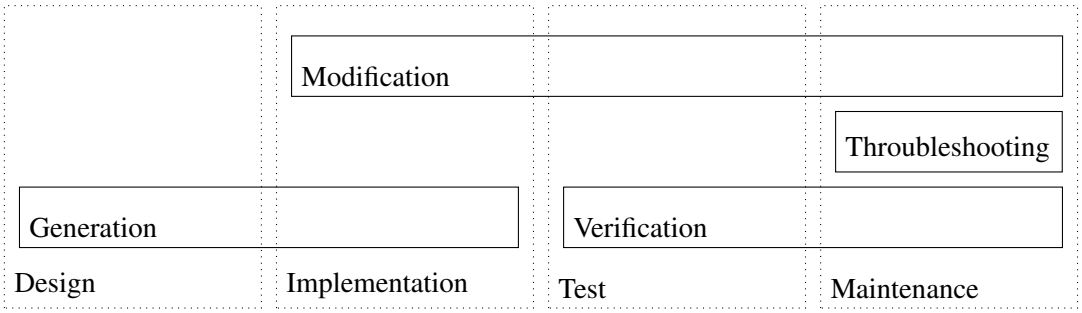


Figure 5.1: Policy analysis workflows

5.1.1 Policy generation workflows

During the policy generation workflow, network administrators translate network security policies from high-level directives into device-specific configurations. Before all security controls are configured and deployed, the different security control configurations must be verified. The first verification to be performed is the anomaly analysis, where potential errors, performance issues and irrelevant rules are identified and removed. Afterwards key aspects of the security directives are verified by performing a reachability analysis. A this point, the network administrator wants to check that essential services are reachable, and unwanted traffic is blocked. Finally, the network administrator desires to compare high-level directives with the device-specific configurations. Therefore, a policy comparison for Implementation-verification is executed.

5.1.2 Policy verification workflows

The policy verification workflow is applied after the security policies have been deployed, and network administrators want to verify that they are working properly. Another use-case for this workflow is when a new attack-path has been discovered, and administrators want to ensure that the network is not vulnerable to it. The main policy analysis technique applied in this workflow is reachability analysis. By executing different reachability queries, network administrators can verify that key security properties are still valid. Should one reachability query not respond with the desired result, one ore more policies must be modified, and the policy modification workflow is applied.

5.1.3 Policy troubleshooting workflows

A common problem in a computer network is that a service is not reachable. Therefore, network administrators want to troubleshoot the connection problem and verify that it is not caused by a policy error. The implemented policies are verified by using the reachability analysis. Should a policy error be the reason for the connection problem, the involved policies are modified. Afterwards, the administrator must ensure that the modification did not introduce new anomalies and that the desired change is applied. Therefore, also, in this case, the policy modification workflow is performed.

5.1.4 Policy modification workflows

There are different reasons why a policy must be modified: a new service or host is deployed, a service or host is removed, or a network vulnerability is found. In all these scenarios, at least, one security policy is modified and after the modification has been performed the new policy must be validated. The validation of a modified policy has several steps, and each step uses a different policy analysis technique. In the first, step anomaly analysis is applied, and it is verified so that no policy anomaly is introduced. The next step is to verify that the desired modification has been performed correctly by performing a reachability analysis. The reachability query is modelled according to the performed modification. In case the result confirms the intention, the next step can be performed, otherwise, the modification must be redone. The last step is performed by using policy comparison for Change-impact-analysis. The administrators verify that the modification does not introduce undesired side effects. The modified policy is compared to the original one, and the comparison algorithm highlights all differences.

5.2 Requirements

The proposed model must follow some essential requirements to be completely functional and future-proof. These requirements are necessary not only for a correct operation but also to ensure an improvement over the state of the art.

The overall solution must guarantee *interoperability* and must be able to work with as many as possible security policies. Since new types of security policies can be introduced in the future, the solution must be *expansible*. Furthermore, the solution must have a well defined *internal format* that can model all required information. Finally, the *representation of results* of the different analysis techniques must be clear to the user and easy to read.

5.2.1 Interoperability

The proposed model should be able to accommodate the most common types of security policies and all possible network topologies. Different network topologies should be modelled by combining various security controls, and security control should be modelled independently from the network topology itself. Security policies should be associated with the security controls in the network topology. The separation between network topology and security policies provides the greatest possible flexibility. Security policies do not need to include topology specific information because they are associated to security controls and not directly to the network topology.

5.2.2 Expansibility

The proposed model should be able to support future security policy types without changing the complete solution. The different analysis techniques should be independent of the input policies and should remain unchanged when a new security policy type is introduced. When a new security policy type is required, only the instantiation of the analysis model should be modified. This means that the impact is reduced because only one modification is required instead of three. Therefore, the impact is minimal, and the support for future security policy types is guaranteed.

5.2.3 Internal format

The solution should model the network topology and all involved network security policies with a common format. This model should also provide the appropriate abstraction between input policies and analysis algorithms. The internal format should enable the

model to compare concrete configurations with a high-level policy. Therefore, it should be possible to populate the internal format with input policies expressed at different abstraction levels. Furthermore, should the internal format be as simple as possible because it is the key enabler for the expansibility requirement.

5.2.4 Representation of results

A correct and complete analysis has no value without a well-defined representation of the results. Therefore, an important part of the overall solution is the result format. It should have a clear structure, and it should be easy to read and interpret. An important consideration is that the main stakeholders are the network administrators. Therefore, the representation of the results should be designed for them and be very familiar.

5.3 Design

The overall model has been designed to incorporate all requirements and to improve the state of the art. It consists of a mathematical model to represent the network topology and the security policies, the internal representation, and one algorithm for each of the three analysis types. The mathematical model used to represent the security policies is called *Geometric-Model*, it uses hyper-rectangles to represent rule conditions. The internal representation, called *Equivalent-Policy*, is computed by a transformation algorithm. The three analysis algorithms use the Equivalent-Policy to perform the analysis. Figure 5.2 shows the different function blocks and how they are interconnected.

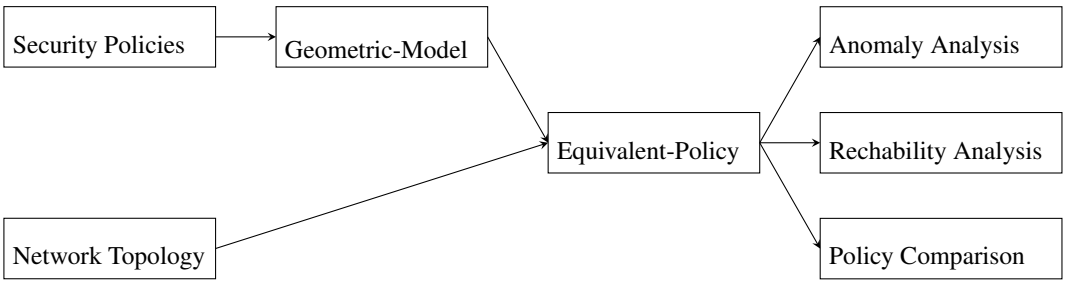


Figure 5.2: Function blocks

5.3.1 Geometric-Model

The Geometric-Model is a mathematical representation of security policies. It is based on the set theory where security policies are represented by a four-tuple consisting of the default action, the rule set, the external data and the resolution strategy. The default action is applied when no rule from the rule set matches a certain packet. The external data is used by the resolution strategy to decided which action to apply when more than one rule matches a packet. A policy rule is composed of the rule condition and the rule action. Rule conditions are represented as hyper-rectangles, where each dimension represents one selector type. Rule relations can be verified by calculating the intersections of the hyper-rectangles.

5.3.2 Equivalent-Policy

The Equivalent-Policy is based on the Geometric-Model and structured the same way as a normal policy. Like a normal policy, it is also composed of a set of rules, a default action, external data and a resolution strategy. It enforces the same action to a packet as the sum of all the other security policies within the network. The Equivalent-Policy is much more efficient in terms of size than a set of policies because fewer rules are required. Although the Equivalent-Policy has fewer rules to evaluate, it still has all the required information to allow the analysis algorithms to retrieve the information about the original rules if necessary. Additionally, it enables an abstraction from the original policies. Therefore, the analysis algorithms must not be modified when a new security policy type is introduced and only the transformation algorithm changes.

5.3.3 Anomaly Analysis

The anomaly analysis algorithm supports Intra-Policy, Inter-Policy and Inter-Technology anomaly analysis. The Intra-Policy anomaly analysis is based on the research performed by Al-Shaer et al. [25] and Basile et al. [37] and has been extended to support a more precise analysis. The Inter-Policy anomaly analysis is also improvement on the works of Al-Shear at al. [26] and includes new anomaly types. A primarily work on Inter-Technology anomaly analysis has been presented in [22]. This work combines all these previous anomaly definitions and proposes a complete solution.

All different anomaly analysis types are performed on the Equivalent-Policy since it contains all the required rule relations. In the case of an Intra-policy analysis, only the rule relations of one policy are required. Therefore, the Equivalent-Policy is filtered, and rule relations must not be recalculated.

5.3.4 Reachability Analysis

The reachability analysis algorithm receives as input the Equivalent-Policy and a reachability query. The reachability query is defined with an improved version of Structured Reachability Query Language (SRQL) [75]. The query result is structured the same way a policy is, and, therefore, well readable by network administrators. The primary work on a reachability analysis algorithm based on the Geometric-Model is presented in [91]. The reachability queries are evaluated by the reachability algorithm on the Equivalent-Policy. The Equivalent-Policy is a compact representation of the global network behaviour and, therefore, ideal for reachability analysis. Furthermore, the Equivalent-Policy contains also all information about the original rules which can be included in the reachability result.

5.3.5 Policy Comparison

The policy comparison algorithm supports all four comparison scopes: Single Policy Change-impact-analysis, Single Policy Implementation-verification, Multiple Policy Change-impact-analysis and Multiple Policy Implementation-verification. The result is in all four cases structured as a policy and is therefore, easily readable by network administrators.

The Equivalent-Policy is best suited for multiple policy comparison because it is a compact representation of the computer network with its security policies. For single policy comparison, the Equivalent-Policy is filtered like in the case of Intra-policy anomaly analysis. The comparison result contains all required information because the Equivalent-Policy contains all information about the original policies.

5.4 Contribution

The main contribution to the state of the art is a new analysis model that can perform all three types of policy analysis. The model and the policy analysis are based on a new concept called Equivalent-Policy. Furthermore, the analysis types are improve the state of the art, and new analysis algorithms are introduced. Finally, the overall approach is more general and can be applied to more scenarios that what has been proposed so far.

The Equivalent-Policy is an innovative concept on how to represent in an efficient way the security behaviour of an entire network. It supports the following security policy types: packet-filter, stateful, Layer7, Routing, IPsec, SSL, NAT/NAPT, Web-Proxies and Monitoring. Such an approach has never been proposed in the state of the art and is, therefore, unique.

The anomaly analysis algorithm has improved and supports new anomaly types in respect to the state of the art. The new anomaly class, called Transformed Traffic Anomaly, occurs when a packet is encrypted or modified and, therefore, can not be evaluated by other policies. The algorithms in the state of the art are designed to evaluate anomalies between rule pairs of the same policy type. The proposed algorithm is designed to identify anomalies also among a set of rules of different policy types. Furthermore, this work supports also more security policies types then the solutions in the state of the art. For the most part, the proposed solutions only support packet-filter policies and IPsec policies.

The reachability analysis algorithm has been improved in three aspects over the state of the art: the query language, the supported security policy types, and the result format. The reachability query language has significantly more features than the solutions proposed in the state of the art. The reachability algorithm supports all types of security policies proposed so far and additionally IPsec policies. The result is structured as a policy and, therefore, well readable. Furthermore, the query result accuracy is introduced, which has never been done before.

The policy comparison algorithm is the first approach that can compare entire network configurations containing different security policies. The comparison algorithms in the state of the only compare single policies of the same type, mainly filtering policies. Furthermore, this model is the first approach that includes all four comparison scopes. This model can perform Implementation-verification and Change-impact-analysis of single policies and entire network configurations. Past research focuses only on one comparison scope, either Change-impact-analysis or Implementation-verification.

The proposed solution will integrate all three policy analysis techniques in one extensible model. This has several advantages, both regarding application but also for future research and development. This is also a significant advantage over the state of the art because a new security policy type needed to be modelled only once but is accessible to three analysis types.

The advantages for the end user are that he has to access and use only one tool for policy analysis. This means that the network topology and the involved security policies must be modelled only once. Furthermore, after the internal model is instantiated the performance of all the analyses is much higher.

The advantages for future research and development are similar and also are based on the fact that only one model exists. By using one internal model and performing all analyses on it, the research effort is reduced. This is because research time is not lost by defining yet another model and it can be used to improve the analysis algorithms of the model itself. Furthermore, it allows the comparison of the performance of different algorithms, because they are executed in the same environment.

Chapter 6

Geometric-Model

This chapter presents the Geometric-Model and how it is used to represent different types of security policies. In particular, the a description for network packets is given. Furthermore, how rule conditions and rule actions are mapped to the Geometric-Model is explained.

6.1 Packets

Before explaining how a security policy and its components are represented by the Geometric-Model, it is important to understand what a security policy is going to inspect, namely network packets. A network packet is a bitstring that is mapped to a *set of fields*. The fields of a packet are for example the source/destination IP address or the source/destination port number. Packet fields in a higher layer include for example the TCP-flags or the HTTP method.

The set of all possible fields is called packet space $\mathbb{P}$ and formally described as $\mathbb{P} = \{\mathbb{F}_1, \ldots, \mathbb{F}_n\}$. The packet space $\mathbb{P}$ is the set of all the possible fields assignments. In general, a packet does not include all possible field but only a subset expressed as $\mathbb{P}_I \subset \mathbb{P}$ indexed by some set $I \subseteq [1, m]$.

$$\mathbb{P}_I = \{\mathbb{F}_i \mid i \in I \subseteq [1, m]\}$$

Furthermore, since fields are disjoint parts of a packet, the union of different fields is the subset of all fields

$$\mathbb{P}_I = \prod_{i \in I} \mathbb{F}_i$$

Packet fields are characterised by their length. Every packet field $\mathbb{F}_i$ is composed of l_i bits thus $\mathbb{F}_i$ is isomorphic to $\left[0{,}2^{l_i} - 1\right] \subset \mathbb{N}$. In the case of packet fields are part of

the packet header the field length is known and constant, whereas the length can also be unknown in other instances. A packet x can be expressed as a set of values, one value for each field. $x = \{f_i \in \mathbb{F}_i,\ \mathbb{F}_i \in \mathbb{P}_I,\ I \subseteq [1,n]\}$, that is $x \in \mathbb{P}_I$, for some set I.

6.2 Conditions

Policy rules define conditions $c = s_1 \times s_2 \times \cdots \times s_\mathrm{m} \subseteq \mathfrak{S}$ to match a packet. The condition c is a set of values s_i, called selector, which represents a set of all possible values for a packet field.

$$c = \prod_{i \in I} s_i \qquad s_i \subseteq \mathbb{F}_i \qquad I \subseteq [1,m]$$

The selection space $\mathfrak{S} = \mathbb{F}_1 \times \mathbb{F}_2 \times \cdots \times \mathbb{F}_\mathrm{m}$ is a subset of packet fields used to define the condition.

$$\mathcal{C} = \prod_{i \in I} \mathbb{F}_i \qquad \mathbb{F}_i \in \mathbb{P}_I \qquad I \subseteq [1,m]$$

A condition c matches a packet $x \in \mathbb{F}_I$ if and only if all selectors s_i match the corresponding packet field of x. A selector s_i matches the corresponding packet field if and only if :

- the packet x contains the corresponding field f_i in $\mathbb{F}_i$
- f_i is in s_i.

For example, a condition $c \in \mathbb{F}_1$ is defined over one selector and the selector is the destination port of the packet. The value assigned to this selector is $s_1 = \{80{,}8080\}$, therefore, there are only two packets that match this condition. The two packets have the value of the packet field $f_1 \in \mathbb{F}_1$ equal to 80 or equal to 8080, where f_1 is the destination port of the packet.

Selectors are linked together by a Cartesian product and form, therefore, a *hyper-rectangle*. Figure 6.1 shows a condition c composed of two selectors $s_1 \in \mathbb{F}_1$ and $s_2 \in \mathbb{F}_2$. The condition c matches packet x_1 but not packet x_2.

6.2.1 Selector representation

Different selector types represent different packet fields. The presented model consists of three selector types: *exact match*, *range-based*, *prefix match* selectors [92] and *regular*

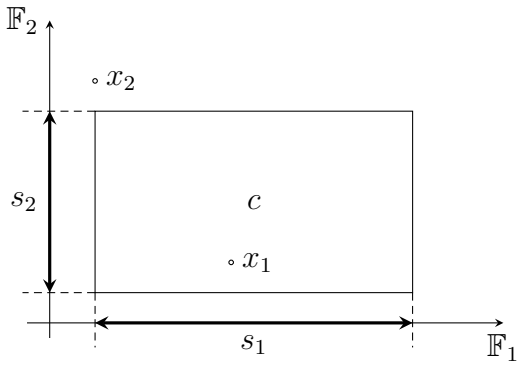


Figure 6.1: Condition $c = s_1 \times s_2$ in a selection space formed by two fields $\mathbb{F}_1$ and $\mathbb{F}_2$

expression match. The elements of an exact match selector are part of a unordered set and can only be compared by equality. The elements of a range-based selector are part of a ordered set and consecutive elements can be grouped, groups are defined by their minimum and maximum element. The elements of a prefix match selector are part of a unordered set, but can be grouped based on a common prefix. The elements of a regular expression match selector are part of a set with an unknown number of elements.

Exact match

Exact match selectors are an unordered set of elements. This is the simplest type of selectors and is best described by the set theory. The intersection between two selectors $s_{i,1}$ and $s_{i,2}$ contains all elements present in both selectors.

$$s_{i,1} = \{\oplus, \ominus, \otimes\}$$

$$s_{i,2} = \{\oslash, \oplus, \ominus\}$$

$$s_{i,1} \cap s_{i,2} = \{\oplus, \ominus\}$$

Range-based

Range-based selectors $s_{i,j}$ consist of groups of consecutively ordered elements. The range is defined by its minimum $s_{i,j_{min}}$ and maximum $s_{i,j_{min}}$ value. The intersection between to range-based selectors $s_{i,1}$ and $s_{i,2}$ is easy to calculate. At least, the minimum value or the maximum element of $s_{i,2}$ is between the minimum and the maximum element $s_{i,1}$.

$$s_{i,1_{min}} < s_{i,2_{min}} < s_{i,1_{max}} \text{ or } s_{i,1_{min}} < s_{i,2_{max}} < s_{i,1_{max}}$$

An example of range-based selectors and their intersection is shown in Figure 6.2. It shows the two selectors $s_{i,1}$ and $s_{i,2}$ which intersect each other. The intersecting set of elements of formed by the minimum element of $s_{i,2}$ and the maximum element of $s_{i,1}$.

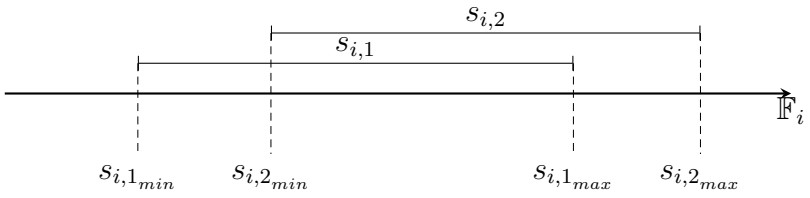


Figure 6.2: Range-based selector.

Regular expression match

Regular expressions are primarily used to define selectors for strings. Although regular expressions are a very compact representation, it is difficult to calculate the intersection between them. The intersection between two regular expressions is best calculated by converting them into deterministic automata [93]. Operations on deterministic automata are much easier and therefore of advantage. The conversion of regular expressions into automata and the intersection between automata is well studied and efficient algorithms are widely available. An example of the intersection of two regular expressions performed using automata is shown in Figure 6.3.

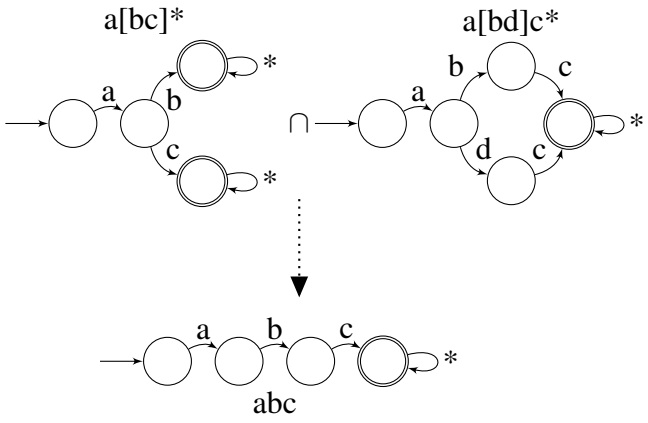


Figure 6.3: Intersection of two deterministic automata.

The conversion from automata into regular expressions, on the other hand, is much more difficult. There are different methods for this, each method has advantages and

disadvantages. The three most common solutions are the transitive closure [94], the algebraic approach (Brzozowski's method) [95], and the state removal [96]. The advantage of the transitive closure is that is fairly simple to implement, but the results are sub-optimal because the created regular expressions are quite long. The advantage of the algebraic approach is that it creates compact regular expressions, but the implementation is very complex. The state removal approach sits between the other two since it is fairly easy to implement and creates reasonable small regular expressions.

Prefix match

Prefix match selectors define the common prefix that all elements in the set start with. Prefix match selectors can be converted into range-based selectors or regular expression match selector. The conversion into range-based selectors is only possible if the dimension of the set is know. The conversion into regular expression match selectors is always possible, however, range-based selectors are preferred because they are more efficient.

For example, a prefix match selector for IP addresses can be converted into a range-based selector because the total number of possible IP addresses is known. The prefix match selector $s = 1.1.1.*$ can be converted into the range-based selector $s' = \{1.1.1.0 - 1.1.1.255\}$ where $s'_{min} = 1.1.1.0$ and $s'_{max} = 1.1.1.255$.

Prefix match selectors for strings, on the other hand, can not be converted into range-based selectors, but in theory only regular expression match selectors because strings are unbounded. The prefix match selector $s = google*$ can be converted into the regular expression match selector $s' = \wedge google \cdot *$.

6.3 Actions

A policy rule is composed a policy action in addition to the policy condition. A policy action is formally defined as a function $a : \mathbb{P} \rightarrow \mathbb{P} \cup \emptyset$ applied to a packet. The function transforms the packet from the packet space $\mathbb{P}$ to the packet space $\mathbb{P} \cup \emptyset$ including the so-called "*null packet*". The null packet is composed of no bits and serves to describe when a packet is dropped.

The action a is a set of values σ_i, called transformations, which represents a set of all modified values for a packet field.

$$\gamma = \prod_{j \in I_{\mathrm{T}}} \sigma_j \qquad \sigma_j \subseteq \mathbb{F}_j \qquad I_{\mathrm{T}} \subseteq [1, m]$$

The transformation space $\mathcal{T} = \mathbb{F}_1 \times \mathbb{F}_2 \times \cdots \times \mathbb{F}_\mathrm{m}$ is a subset of packet fields used to define the action.

$$\mathcal{T} = \prod_{j \in I_\mathrm{T}} \mathbb{F}_j \qquad \mathbb{F}_j \in \mathbb{P} \qquad I_\mathrm{T} \subseteq [1, m]$$

The action set $\mathcal{A} = \{a : \mathbb{P} \to \mathbb{P} \cup \emptyset\}$ includes all possible actions. Examples of valid actions are: ALLOW and DENY in case of firewalls, MONITOR in case of network monitors, REDIRECT in case of routers, MODIFY in case of NATs, NEW HEADER in case of IPsec AH, ENCRYPT in case of IPsec ESP. These actions can be modelled by means of the action function a as follows:
The ALLOW action applies the *identity function* to the packet and leaves the packet therefore unchanged:

$$\texttt{ALLOW} : \mathbb{P} \to \mathbb{P} \cup \emptyset \text{ such that } x \mapsto x$$

The MONITOR action is very similar to the ALLOW action because the packet remains unchanged and therefore it also applies the *identity function*:

$$\texttt{MONITOR} : \mathbb{P} \to \mathbb{P} \cup \emptyset \text{ such that } x \mapsto x$$

The DENY action blocks a packet and therefore maps it to the *null packet*:

$$\texttt{DENY} : \mathbb{P} \to \mathbb{P} \cup \emptyset \text{ such that } x \mapsto \emptyset$$

The REDIRECT action forwards a packet to a certain network interface and therefore can be maps the packet to the *identity function* in case of the forwarded interface $\mathcal{I}_{FW}$ and to the *null packet* for all other interfaces:

$$\texttt{REDIRECT} : \mathbb{P} \to \mathbb{P} \cup \emptyset \text{ such that } x \mapsto \emptyset \; \forall \, \mathcal{I}_n \neq \mathcal{I}_{FW} \text{ and } x \mapsto x \; \forall \, \mathcal{I}_n = \mathcal{I}_{FW}$$

The MODIFY action changes the value of certain packet fields, therefore the action clause contains the fields and the new values to assign:

$$\texttt{MODIFY} : \mathbb{P} \to \mathbb{P} \cup \emptyset \text{ such that } x \mapsto x_a$$

The NEW HEADER action encapsulates a packet in a new header and can be modelled such as the original header is overwritten by the new one, therefore the action clause all packet fields of the new header and the assigned values:

$$\texttt{NEWHEADER} : \mathbb{P} \to \mathbb{P} \cup \emptyset \text{ such that } x \mapsto x_a$$

The ENCRYPT action encrypts either the entire payload of a specific layer or only a single packet field The action clause contains the layer (layer 4 PL_4, layer 7 PL_7) or the packet field that will be encrypted. This action is modelled by setting the value of the encrypted packet field to 0:

$$\texttt{ENCRYPT} : \mathbb{P} \rightarrow \mathbb{P} \cup \emptyset \text{ such that } x \mapsto x_0$$

6.4 Resolution strategy

The resolution strategy decides which rule to enforce when more than one rule matches a packet. It takes its decision based on three different criteria: condition clause, action clause and external data. An example for a resolution strategy that takes its decision based on the condition clause is the *Most Specific Takes Precedence* (MSTP), more specific rules prevail over more general rules. An example for a resolution strategy that takes its decision based on the action clause is the *Deny Takes Precedence* (DTP), the DENY action is applied when at least one rule enforces DENY. An example for a resolution strategy that takes its decision based on an external data is the *First Matching Rule* (FMR), the rule with the highest priority prevails.

Because the condition clause and the action clause is part of the rule itself, it is easy to model a resolution strategy based on them. The external data, however, is not part of the rule and must be associated first to each rule before a resolution strategy can be applied. The association of external data to a rule r_i can be expressed by the external data function ε_E:

$$\varepsilon_E(r_i) = (r_i, f_1(r_i), f_2(r_i), f_3(r_i), \ldots)$$

where E is a set of functions that associate rules with external attributes X_j

$$E = \{f_j : R \rightarrow X_j\}_j$$

The external data function ε_E is composed with the external data resolution strategy $\mathfrak{R}_E$ resulting in a resolution function $\mathfrak{R}$ able to handle all three decision criteria:

$$\mathfrak{R} = \mathfrak{R}_E \circ \varepsilon_E$$

that is,

$$\mathfrak{R} : \quad \{r_l, r_m, \ldots\} \quad \overset{\varepsilon_E}{\longmapsto} \quad \{\varepsilon_E(r_l), \varepsilon_E(r_m), \ldots\} \quad \overset{\mathfrak{R}_E}{\longmapsto} \quad a$$

6.5 Policies

Security policies are composed of policy rules, the resolution strategy, optional external data and a default action. Formally policies are represented by the four-tuple $(R, \mathfrak{R}, E, a_d)$. This representation is independent of the applied resolution strategy. Should the resolution strategy not require external data, the set E is empty, and the policy is represented as $(R, \mathfrak{R}, \emptyset, a_d)$. The single components of the policy model are:

- $R = \{r_i\}_i$, $i \in [1, n]$ represents the set of rules, where each rule is an association $r = (c, a)$ of the condition clause $c \subseteq \mathcal{C}$ and a the action $a \in \mathcal{A}$
- $\mathfrak{R} : 2^R \rightarrow \mathcal{A}$ represents the resolution strategy, it returns the action to enforce in case more than one rule matches a packet.
- $E = \{E_1, E_2, \ldots\}$ represents the optional external data, the external data functions $\varepsilon_k : R \rightarrow E_k$ associates the external data to the rules.
- a_d represents the default action.

The policy itself can be modelled as a function from the selection space $\mathfrak{S}$ to the action space $\mathcal{A}$.

$$p : \mathfrak{S} \rightarrow \mathcal{A}$$

It applies the appropriate action to a packet x based on the rule set, the resolution strategy and the default action. The first step is to identify the rules from the rules set where the packet x satisfies the rule condition. This step is called rule matching and performed by the matching function match_R. The matching function returns either an empty set, a single rule or a set of rules. In case, the set contains more than one rule the resolution strategy $\mathfrak{R}$ is applied to the set

$$\mathfrak{R} : 2^R \rightarrow \mathcal{A}$$

that decides which action to enforce. If the set is empty, the action to enforce is defined by the default action a_d. Therefore, a policy can be modelled as follows:

$$p(x) = \begin{cases} d & \text{if } \mathtt{match}_\mathtt{R}(x) = \emptyset \\ a_i & \text{if } \mathtt{match}_\mathtt{R}(x) = \{r_i\} \\ \mathfrak{R}(\{r_l, r_m, \ldots\}) & \text{if } \mathtt{match}_\mathtt{R}(x) = \{r_l, r_m, \ldots\} \end{cases}$$

However by defining $\mathfrak{R}(\emptyset) = d$ and $\mathfrak{R}(r_i) = a_i$, the policy p can be simplified as:

$$p(x) = \mathfrak{R}(\mathtt{match}_R(x))$$

6.5.1 Policy operations

A policy can perform different operations on the rule set to evaluate its properties and identify specific rules. Additionally to the matching function, already introduced in the previous section, a policy can also perform the effective function and the effective cover function. The effective function identifies the part of the condition clause which effectively enforces an action. The effective cover function identifies the action that is covered by the effective part of the condition clause.

Matching function

When a policy must enforce an action to a packet, it first must identify all rules that are defined for the packet. The matching function match_R returns the subset M of rules in R whose conditions match the packet x.

$$\begin{aligned} \mathrm{match}_R : \quad \mathfrak{S} &\longrightarrow 2^R \\ x &\longmapsto M = \{r_i \in R \mid x \in c_i\} \end{aligned}$$

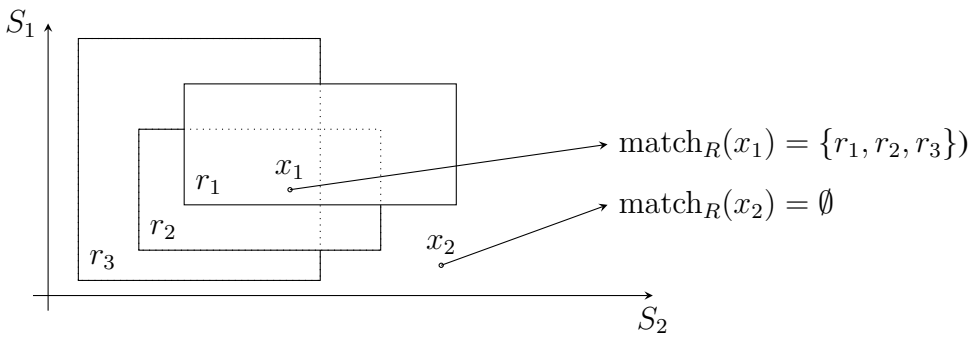


Figure 6.4: Example of match_R

Figure 6.4 shows an example of the matching function match_R applied to the rule set R for two packets x_1 and x_2. The packet x_1 is matched by the rules r_1, r_2 and r_3. Therefore, the matching function returns a set containing these three rules $\mathrm{match}_R(x_1) = \{r_1, r_2, r_3\}$. The packet x_2 is not matched by any rule and, therefore, the matching function returns an empty set $\mathrm{match}_R(x_2) = \emptyset$.

Effective Function

Another important evaluation the policy model can make is to calculate the part of a rule that effective will be enforced. The simplest method to evaluate this is to eliminate all

parts of the condition clause which are covered by other rules. The remaining part is what effectually enforces an action and if the rule is completely covered nothing remains. Formally this algorithm is applied by the effective function $\mathrm{eff}_p(r)$:

$$\begin{aligned} \mathrm{eff}_p: \quad R &\rightarrow & 2^{\mathfrak{S}} \\ r &\mapsto & x \in c \text{ such that } \mathfrak{R}(\mathrm{match}_R(x)) \neq \mathfrak{R}(\mathrm{match}_R(x) \setminus \{r\}) \end{aligned}$$

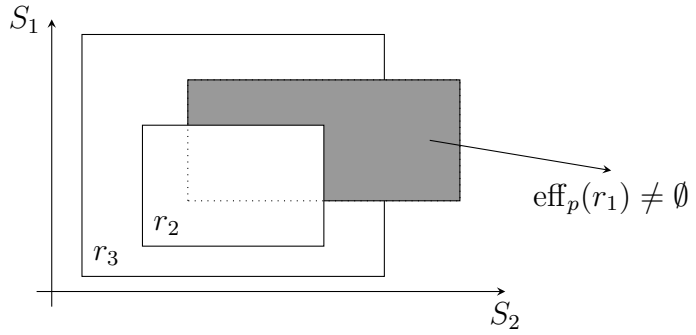


Figure 6.5: The $\mathrm{eff}_p(r_1)$ is represented by the grey part

Figure 6.5 shows an example of the effective function applied to rule r_1. Rule r_1 is partly covered by rule r_2 and covers rule r_3. The part of rule r_1 that effectually is enforced ($\mathrm{eff}_p(r_1)$) is $\mathrm{eff}_p(r_1) = c_1 \setminus c_2$ and highlighted in gray. It is worth mentioning that in general it is not possible to represent $\mathrm{eff}_p(r_1)$ as a single hyper-rectangle. In this example, it is represented as the relative complement of two conditions clauses. However, it is not required to perform this computation.

Effective Cover Function

The counter part of the effective function is the effective cover function $\mathrm{effc}_p(r)$. This function evaluates which rules covered by the effective part of a rule r enforce a different action. This can also be described as the parts of the decision space that change the enforced action in case the rule is deleted. The simplest method to evaluate this is to eliminate al parts from the effective parts that cover other rules that enforce the same action. The remaining part is what effectually enforces an action and does not cover any other effective part with the same action. If the rule is completely covered or the effective part covers completely the same actions, nothing remains.

$$\begin{aligned} \mathrm{effc}_p: \quad R &\rightarrow & 2^{\mathfrak{S}} \\ r &\mapsto & x \in c \text{ such that } \mathfrak{R}(\mathrm{match}_R(x)) \neq \mathfrak{R}(\mathrm{match}_R(x) \setminus \{r\}) \end{aligned}$$

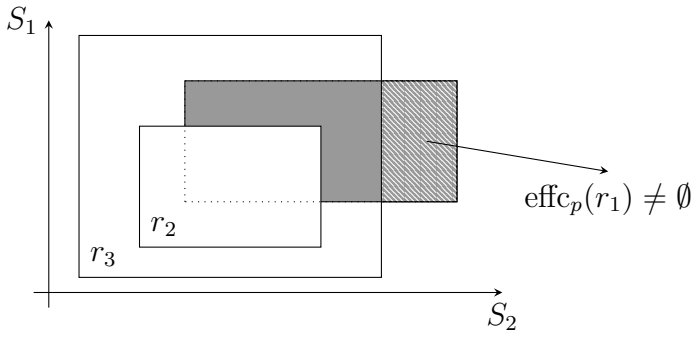


Figure 6.6: The $\mathrm{effc}_p(r_1)$ is represented by the grey/white pattern part

Figure 6.6 shows an example of the effective cover function applied to rule r_1. This example uses the same rules and rule priority as the previous one (Figure 6.5). All rules enforce the same action, and the default action is different than the action enforced by the rules. Rule r_1 is partly covered by the rules r_2 and covers rule r_3, the remaining part ($\mathrm{eff}_p(r_1)$) is highlighted. The effective cover function $\mathrm{effc}_p(r_1)$ is highlighted by the grey/white pattern, it is $\mathrm{eff}_p(r_1) \setminus r_3$. Rule r_3 enforces the same action as rule r_1 does, and therefore, this part will not alter the enforced action. The remaining part $\mathrm{effc}_p(r_1)$, however, will, because the default action is different than the action of rule r_1.

Chapter 7

Equivalent-Policy

This chapter presents how the Equivalent-Policy is calculated. The Equivalent-Policy is a black-box representation of the entire network that summarizes its end-to-end security behaviour. The calculation is subdivided into three steps. Firstly, all transformation rules are processed to represent all changes applied to the packet fields in the downstream security policies. Secondly, all rule relations and the resulting policy action is calculated. Thirdly, the minimum number of rules is exported as the Equivalent-Policy.

7.1 Transformation Resolution

The first step when calculating the Equivalent-Policy is to resolve all transformation applied by the action MODIFY and NEW HEADER. The transformation resolution is required only by these two actions because they modify the packet structure. Therefore, the linear evaluation with other downstream rules is not possible. After the transformation, the ability for linear evaluation is restored, and the analysis processes have a much better performance.

The rule action clause γ_i defines which transformations must be applied. The transformation resolution selects all rules $r = (c, a)$ from the rule set R where the rule condition $c = s_1 \times s_2 \times \cdots \times s_m$ intersects the action clause γ_i. For each of them, it creates a new rule $r^{\mathrm{T}} = \left(c^{\mathrm{T}}, a\right)$ where $c^{\mathrm{T}} = s_1^{\mathrm{T}} \times s_2^{\mathrm{T}} \times \cdots \times s_m^{\mathrm{T}}$ and each $s_i^{\mathrm{T}} \subseteq \mathbb{F}_i$ is obtained as follows:

- $s_i^{\mathrm{T}} = c_i \subseteq \mathbb{F}_i$ from c_i if $\mathbb{F}_i$ is in $\mathcal{T}$;
- $s_i^{\mathrm{T}} = s_i$ if $\mathbb{F}_i$ is not in $\mathcal{T}$.

For a formal definition of this operation some additional functions are required:

- *transformation-condition extraction* : $\mathrm{in}(\tau_i) = c$;
- *transformation-action extraction* : $\mathrm{out}(\tau_i) = \gamma_i$;
- *projection on transformation space* : $\Psi\left(r_i\right) = \prod_{j \in I_\mathrm{T}} s_{ij}$;
- *rule condition extraction* : $\mathcal{C}(r_i) = c_i$.

The new rules created by the *rule transformation function* Θ are inserted in the downstream policy. The transformation function returns only a new rule $r_i^{(j)}$ if its condition clause c_i intersects the transformation-action clause.

$$\Theta(r_i, \tau_j) : r_i \longmapsto r_i^{(j)}$$

where $r_i^{(j)}$ is defined as:

$$r_i^{(j)} = \begin{cases} (\Sigma(c_i, \mathrm{in}(\tau_i)), a_i) & \text{if } \Psi\left(r_i\right) \cap \mathrm{out}(\tau_j) \neq \emptyset \\ \emptyset & \text{if } \Psi\left(r_i\right) \cap \mathrm{out}(\tau_j) = \emptyset \end{cases}$$

By abuse of notation, $r_i = \Theta^{-1}\left(r_i^{(j)}\right)$ if $r_i^{(j)} = \Theta(r_i, c)$ for some τ_j. The new rules inherit the values of the external data from the original rules:

$$\forall k, \varepsilon_k(r_i^{(j)}) = \varepsilon_k(r_i)$$

A special case to consider is when a transformed packet is not matched by any rule. This means that the default action must be applied, and the transformation resolution must operate accordingly. This requirement is ensured by the transformation resolution that creates an additional rule called *default transformation rule*. It enforces the default action and has the condition clause equal to the transformation-action clause. Formally this it is defined as $\forall r_i, \Psi\left(r_i\right) \cap \mathrm{out}(\tau_j) = \emptyset$ and the transformation rule $r^{(\tau_j)}$ is created as follows:

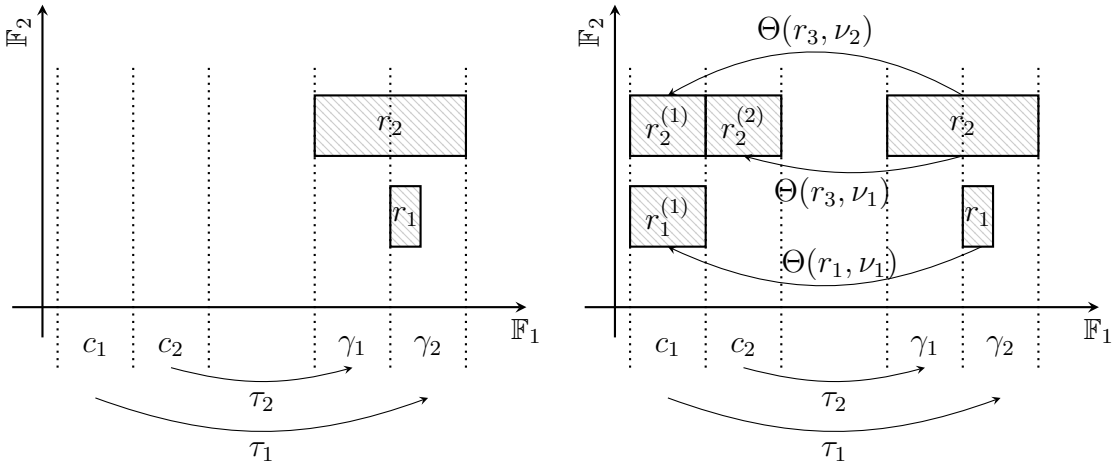
$$r^{(\tau_j)} = (\Sigma(\mathbb{F}_I, in(\tau_j)), a_d)$$

The set of all default transformation rules $r^{(\tau_j)}$ is named $D^{(T)}$ and has the highest priority. The cardinality of $D^{(T)}$ is always less than or equal to the cardinality of T.

In summary, the transformation resolution calculates $D^{(T)}$ and applies $\Theta(r_i, \tau_j)$ to all rules $r_i \in R$ and transformation actions $\tau_j \in T$:

$$R^{(T)} = \{\Theta(r_i, \tau_j)\}_{i,j} \cup D^{(T)}, \text{ with } i \leq m, j \leq t$$

Figure 7.1 shows an example of a policy before and after the transformation resolution is applied. Figure 7.1a shows the original policy that is composed by two rules $\{r_1, r_2\}$. There are two transformation rules defined $rt_1 = (c_1, \gamma_1)$ and $rt_2 = (c_2, \gamma_2)$. rt_1 maps the x-interval c_1 into γ_1 and rt_2 maps c_2 into γ_2. The first rule r_1 covers the entire transformation-action γ_1. The second rule r_2 covers both transformation-action γ_1 and γ_3. Figure 7.1b shows the resulting policy after the transformation resolution. Rule $r_1^{(1)}$ is added by the transformation function $\Theta(r_1, \tau_1)$. Rule $r_2^{(1)}$ is added by the transformation function $\Theta(r_2, \tau_1)$. Rule $r_2^{(2)}$ is added by the transformation function $\Theta(r_2, \tau_2)$.



(a) A policy p with rules intersecting some γ.

(b) The policy p after the transformation.

Figure 7.1: Composition of transformation and filtering policies.

After applying a transformation resolution to a policy $p = (R, \mathfrak{R}, E, d)$ a new T-modified policy $p^{(T)}$ is created:

$$p^{(T)} = (R \cup R^{(T)}, \mathfrak{T}, E, d)$$

The T-modified policy uses the *T-modified resolution strategy* $\mathfrak{T}$:

$$\begin{array}{ccc} \mathfrak{T} : 2^{R \cup R^{(T)}} & \rightarrow & \mathcal{A} \\ M_{\text{eq}} \subseteq R \cup R^{(T)} & \longmapsto & \begin{cases} \mathfrak{R}(H) & \text{if } M^{(T)} \neq \emptyset \\ \mathfrak{R}(M) & \text{if } M^{(T)} = \emptyset \end{cases} \end{array}$$

where $M_{\text{eq}} = \text{match}_{R \cup R^{(T)}}(x) = M \cup M^{(T)}$, $M \subseteq R$, $M^{(T)} \subseteq R^{(T)}$ and $H = \{\Theta^{-1}\left(r_i^{(j)}\right) \mid r_i^{(j)} \in M^{(T)}\} \subseteq R$.

The T-modified resolution strategy can calculate a result for any subset of $R \cup R^{(T)}$ and is, therefore, sound. The reason is that it uses the original resolution strategy $\mathfrak{R}$ and

the external data inherited form the original policy. This assumption is also valid for resolution strategies, such as FMR, that require unique value external data. The transformation resolution may introduce duplicate external data, but it is never used at the same time because transformed rules and the original ones will never intersect.

7.1.1 Multiple transformations

A packet from source to destination may encounter multiple transformation rules, and therefore, the transformation resolution must be applied multiple times. For example, on the path from source to destination are two transformation policies T_1 and T_2 before another security policy p. The security policy p will be (T_1, T_2)-modified recursively and result in $p^{(T_1,T_2)}$:

$$p^{(T_1,T_2)} = \left(p^{(T_1)}\right)^{(T_2)}$$

and the resulting rule set is:

$$R \cup R^{(T_1)} \cup \left(R \cup R^{(T)}\right)^{(T_2)} = R \cup R^{(T_1)} \cup R^{(T_2)} \cup R^{(T_2,T_1)}$$

The (T_1, T_2)-modified resolution strategy operates in stages, first $R^{(T_2,T_1)}$, then $R^{(T_2)}$, then $R^{(T_1)}$ and finally R.

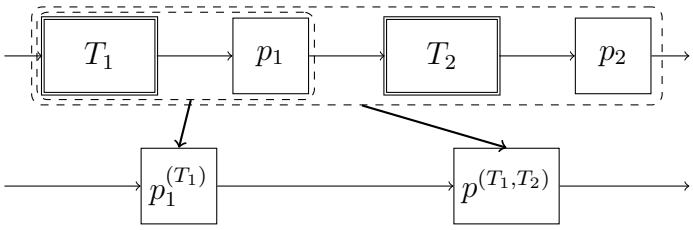


Figure 7.2: Multiple transformation resolution of policy p.

Figure 7.2 shows an example of four in series connected security policies, two policies (T_1, T_2) that require a transformation resolution and two (p_1, p_2) that do not. After the resolution transformation is applied, only two policies remain:

- $p_1^{(T_1)}$ is the T_1-modified policy p_1 (i.e., the inner dashed rectangle);
- $p_2^{(T_1,T_2)}$ is the (T_1, T_2)-modified p_2 (i.e., the external dashed rectangle).

7.1.2 Inverse transformations

When a packet is encapsulated in a tunnel two transformations are applied, one when it enters the tunnel and one when it exits. The first transformation resolution is already applied, but the inverse one must be explicit modelled. Therefore, for the inverse cases, all security policies in the path are pre-processed and such cases are identified. When an inverse case is identified the corresponding transformation rule is removed from future transformation resolutions.

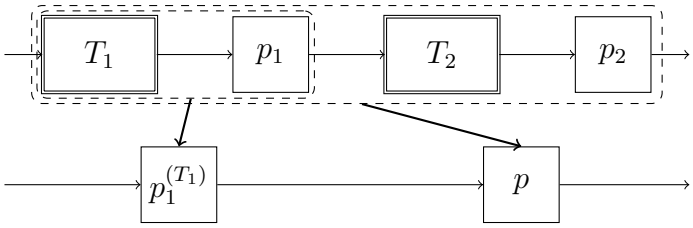


Figure 7.3: Inverse transformation resolution of policy p.

Figure 7.3 shows an example of four in series connected security policies, two policies (T_1, T_2) that require a transformation resolution and two (p_1, p_2) that do not. In particular, the T_2 applies the inverse transformation of T_1 and therefore the policy p_2 remains unchanged. After the resolution transformation is applied, only two policies remain

7.2 Canonical From

After all transformation rules have been substituted the Equivalent-Policy can be calculated. The Equivalent-Policy relies only on set operations to represent all rules. The first step when calculating the Equivalent-Policy is to compose all correlated rules and the interconnected policies. The result of this operation is the so-called *Canonical Form*.

Rule composition

The rule composition is applied when two or more rules are correlated, which means that their condition clauses intersect. It enforces the action calculated by means of the resolution strategy of the involved policies.

The rule composition "$\circ$" of k rules $r_i \in R$ in a policy p is formally defined as:

$$\begin{array}{rccc} \circ: & R \times R \times \cdots \times R & \rightarrow & k^{\mathfrak{S}} \times \mathcal{A} \\ & r_{1,2,\cdots,k} = r_1 \circ r_2 \circ \cdots \circ r_k & \mapsto & (c_1 \cap c_2 \cap \cdots \cap c_k, \mathfrak{R}(\{r_1, r_2, \ldots, r_k\})) \end{array}$$

The set of all possible compositions of rules in R is called closure $\overline{R}$:

$$\forall\, r \in R \Rightarrow r \in \overline{R} \quad \wedge \quad \forall\, r_1, r_2 \in \overline{R} \Rightarrow r_1 \cdot r_2 \in \overline{R}$$

The closure $\overline{R}$ also contains rules with equivalent condition clauses and, therefore, redundant information. R^* is defined as the subset of $\overline{R}$ that contains unique condition clauses. When two or more rules in $\overline{R}$ have the same condition clause only the rule composed by the greatest number of rules is selected.

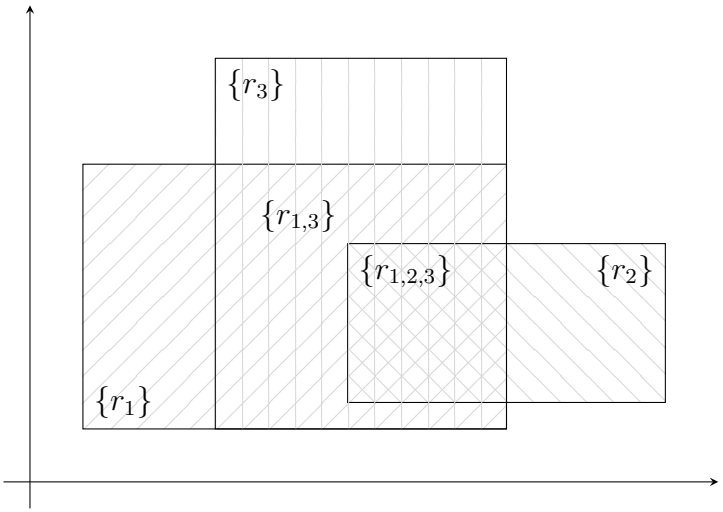


Figure 7.4: R^* of policy $(R = \{r_1, r_2, r_3\}, \mathfrak{R}_E, E, d)$

Fig. 7.4 represents a policy containing three rules r_1, r_2 and r_3. The closure $\overline{R}$ of this policy is composed of $\{r_1, r_2, r_3, r_{1,3}, r_{1,2}, r_{2,3}, r_{1,2,3}\}$ where the intersecting rules $r_{1,2}, r_{2,3}, r_{1,2,3}$ are equivalent. Therefore R^* only contains the rules $\{r_1, r_2, r_3, r_{1,3}, r_{1,2,3}\}$ but not $r_{1,2}, r_{2,3}$.

R^* has the following properties:

- all rules in R^* have distinct condition clauses;
- given two intersecting rules $r_1, r_2 \in R^*$, there exists a rule in R^* whose condition clause is the intersection of the condition clauses of r_1 and r_2, that is, there exists a rule $r = (c_1 \cap c_2, a)$ (*closure with respect to the intersection*).

It is possible to define an order relation "$\leq$" for R^* based on the dimension of the condition clauses:

$$r_1, r_2 \in R^*, r_1 \leq r_2 \iff c_1 \supseteq c_2$$

The structure $(R^*, \leq)$ is a partially ordered set with respect to $\leq$ and, therefore, a semi-lattice. Furthermore, the least upper bound of $\{r_1, r_2\} = (c_1 \cap c_2, a)$ is contained in R^* due to the closure with respect to the intersection.

The composition of rules with conditions containing different selectors is also possible. It is sufficient to add the missing selectors to each rule and assume its value tautological, that is, the value will be as big as the entire selector. This operation is called *decision space homogenisation*.

Policy composition

The policy composition is applied when the Equivalent-Policy of two or more interconnected security controls is calculated. An important property of interconnected security controls is that they do not interact directly with each other. There is no information flow between them, a downstream policy receives the packets from the upstream policy but does not know what action have been applied.

Security controls enforce different actions, but a packet can arrive at its destination only if no policy on the path enforces the DENY action. The *serial composition* function models these interactions between policies:

$$+ : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$$

it describes the composition of actions of two in series connected security controls:

$$a + a = a$$
$$a + d = d + a = d + d = d$$

The result of the serial composition is ALLOW only when both actions are ALLOW.

Two policies $p_1 = (R_1, \mathfrak{R}_1, E_1, a_{d1})$ and $p_2 = (R_2, \mathfrak{R}_2, E_2, a_{d2})$ are composed by applying the serial composition $+$. The serial composed policy $p_1 + p_2$ is defined as $(R_1 \cup R_2, \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}, \emptyset, a_{d1} + a_{d2})$, where the default action is the serial composition of the two default actions and the composed resolution strategy $\mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}$ is defined as:

$$\begin{array}{rccc} \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}: & 2^{R_1 \cup R_2} & \longrightarrow & \mathcal{A} \\ & S_1 \cup S_2 & \longmapsto & \mathfrak{R}_1(S_1) + \mathfrak{R}_2(S_2) \end{array}$$

The serial composition is an associative operation and can, therefore, be applied also to more than two in series connected security controls. For example, if there are three in series connected security controls the composition of all three is $(p_1 + p_2) + p_3 = p_1 + (p_2 + p_3)$ represented as $(R_1 \cup R_2 \cup R_3, \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2,\mathfrak{R}_3}, a_{d1} + a_{d2} + a_{d3})$.

7.2.1 Canonical Form Calculation

Based on the order relation "$\leq$" of R^* a new resolution strategy called *canonical form* (CAN) is defined. The canonical form decides which action to enforce based on the least upper bound rule in R^*:

$$\begin{array}{lrcll} \mathrm{CAN}: & 2^{R^*} & \rightarrow & \mathcal{A} & \\ & \{r_l, r_m, \ldots\} & \mapsto & a & \text{such that } r = (c, a) = \\ & & & & = \underset{R^*}{\mathrm{lub}}\{r_l, r_m, \ldots\} \end{array}$$

A policy $p_{\mathrm{CAN}} = (R^*, \mathrm{CAN}, \emptyset, d)$ is called canonical form policy when R^* is calculated from the rule-set R of another policy $p = (R, \mathfrak{R}_E, E, d)$. The canonical form policy is semantically equivalent to the original policy, but often has a greater number of rules. The main advantage of the canonical form is that has all possible rule compositions pre-computed. Therefore, any algorithm working on the canonical form does not need to consider them any more and is independent of the original policy and its resolution strategy.

Different security policies on a path have many similar rules, the reason being to allow end-to-end communication on a path all security controls must enforce the ALLOW action. This property is of advantage for the creation of the Equivalent-Policy because duplicate rules are combined into one and, therefore, fewer rules are required. Furthermore, all rules that do not concern the source and the destination of the path can also be eliminated.

7.3 Semantic preserving morphism

Starting from the semi-lattice representation of the canonical form a policy with a generic resolution strategy can be extracted. The policy extraction algorithm is called semantic preserving morphism.

This section presents the semantic preserving morphism for FMR-policies, the so-called FMR-morphism. The FMR-morphism extracts from the canonical form policy $(R^*, \mathrm{CAN}, \emptyset, d)$ a set of rules R_{FMR} and associates to each rule of this set a priority $\pi : R_{\mathrm{FMR}} \rightarrow \mathbb{N}$. The algorithm will map the partially ordered set (semi-lattice) to a graph, the so-called *cover graph*. The graph representation is optimal for selecting the required rules and associating them a priority.

The cover graph of a partially ordered set $(S, \leq)$ is a directed acyclic graph. The vertices of the graph represent the elements in S, and the edges of the graph represent the

cover relation. Since every policy p is representable as a partially ordered set $(R^*, \leq)$ by the canonical form, it is also possible to represent it as a cover graph $G(p)$. Consequently, the R^* always contains a top element T which is greater than all others, even if $c_1 \cap c_2 \cap \cdots \cap c_n = \emptyset$ where n is the number of rules in R.

$$\begin{aligned} T &= \operatorname*{lub}_{r \in R^*} \{r\} = r_1 \cdot r_2 \cdot \ldots \cdot r_n = \\ &= (c_1 \cap c_2 \cap \cdots \cap c_n, \mathfrak{R}\{r_1, r_2, \ldots, r_n\}) \end{aligned}$$

After the cover graph has been created the necessary rules can be extracted and priorities assigned to them. For example, if R^* contains two distinct conflicting rules r_x and r_y such that $r_x \leq r_y$ and $\mathrm{CAN}\{r_x, r_y\} = a_y$. This means r_y has the higher order in R^*, and the FMR-morphism must assign a higher priority to rule r_y. In general, this is also valid for the least upper bounds:

$$\pi(\operatorname*{lub}_{R^*}\{r_1, r_2\}) < \pi(r_1) \quad \wedge \quad \pi(\operatorname*{lub}_{R^*}\{r_1, r_2\}) < \pi(r_2)$$

The algorithm that selects the rules and their priorities is based on a modified breadth-first backward traversal in $G(p)$ and all unnecessary rules are discarded. In order to discard a rule, its condition clause must be strictly included in one of its parents or ancestors. To verify this condition the concept of *maximal domain* $\mathcal{M}(r)$ associated with a rule r, named the base rule, is introduced. The maximal domain is a sub-graph of $G(p)$ and includes only the required rules. The reduced maximal domain is the subset $M^* \subseteq M$ that contains the rules whose ancestors are not in M. All the condition clauses of rules in M are subsets of at least one rule in the reduced maximal domain.

Formally, the maximal domain $\mathcal{M}(r)$ of a base rule $r \in G(p)$ is the set computed as follows:

1. (*base clause*) $r \in \mathcal{M}(r)$;
2. (*inductive clause*) if $r'' \in G(p)$ and $r' \in \mathcal{M}(r)$ are such that $r'' \prec r'$, $a_{r''} = a_r$ and all the rules in every path between r'' and r enforce the same action a_r, then $r'' \in \mathcal{M}(r)$

The maximal domain is based on two statements, the base clause and the inductive clause. The base clause requires that the rule is in the maximal domain. The inductive clause requires that in order to substitute r with one of its ancestors r_a, all rules on the path from r to r_a enforce the same action. The ancestor r_a is suitable because it is never overwritten by a different action of a covering rule. Therefore, r is unnecessary in the FMR representation and can be discarded.

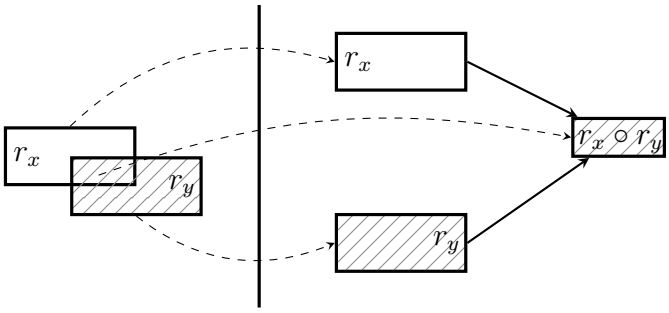
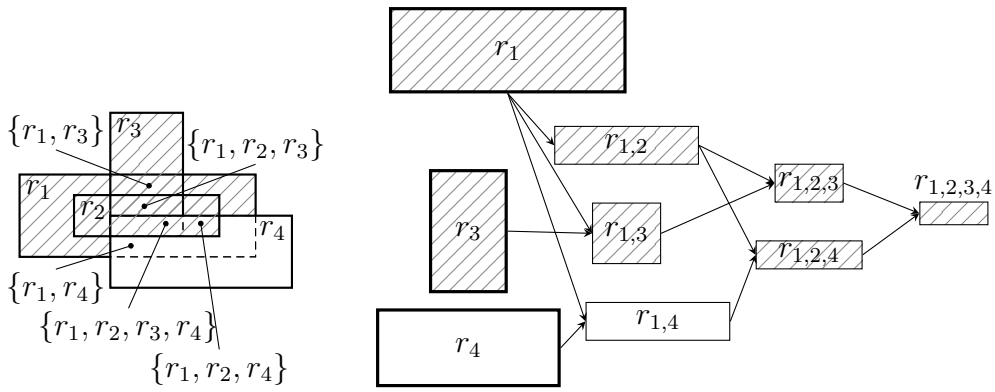


Figure 7.5: r_y can substitute $r_x \cdot r_y$ as it enforces the same action.

Figure 7.5 shows an example of a maximal domain where the base rule is $r_x \circ r_y$ and its ancestor is r_y. Both rules enforce the same action and therefore according to the definition of maximal domain can be discarded. To calculate all maximal domains of an entire cover graph $G(p)$, first top element is used as base rule and then sequentially all rules not part of a maximal domain are evaluated.



(a) Geometric representation. (b) canonical form semi-lattice

Figure 7.6: Usage of the proposed FMR-morphism to sample policy p.

Figure 7.6 shows an example of the maximal domain of a policy. Figure 7.6a represents the geometric representation of a policy whose canonical form semi-lattice is represented in Figure 7.6b. The maximal domains of the semi-lattice are highlighted with circles in Figure 7.6b. The first base rule is $r_{1,2,3,4}$ and its resulting maximal domain is:

$$M_1 = \mathcal{M}(r_{1,2,3,4}) = \{r_{1,2,3}, r_{1,2,4}, r_{1,2}, r_{1,3}, r_3\}$$

At this point the remaining rules to evaluate are $r_{1,4}$, r_4 and r_1. $r_{1,4}$ and r_4 are not in the maximal domain M_1 because the enforce a different action. r_1 is not included because the

path from the base rule $r_{1,2,3,4}$ to r_1 includes also $r_{1,4}$ which has a different action. The next base rule to evaluate is $r_{1,4}$ and its resulting maximal domain is:

$$M_2 = \mathcal{M}(r_{1,4}) = \{r_4, r_{1,4}\}$$

Finally, the remaining base rule to evaluate is r_1 and its resulting maximal domain is the rule itself:

$$M_3 = \mathcal{M}(r_1) = \{r_1\}$$

After all maximal domains have been calculated, the irrelevant rules from the single domains are removed:

$$M_1^* = \{r_3, r_{1,2}\} \text{ and } M_2^* = \{r_4\} \text{ and } M_3^* = \{r_1\}$$

The rule set R_{FMR} is the union of all maximal domains obtained:

$$R_{FMR} = M_1^* \cup M_2^* \cup \cdots \cup M_x^*$$

The priorities of the single rules are assigned according to the order in the reduced cover graph, $\forall r_i \in M_i^*$ and $\forall r_j \in M_j^*$, $i < j$ implies that $\pi(r_i) < \pi(r_j)$. The relative priorities in one maximal domain are not important since all rules enforce the same action. Therefore, assigning priority 1 to rule r_3 ($\pi(r_3) = 1$) and priority 2 to rule 1,2 ($\pi(r_{1,2} = 2)$ or vice versa are both valid. Table 7.1 shows the equivalent FMR policy obtained from the example policy.

Rule	**Priority**
r_3	$\pi(r_3) = 1$
$r_{1,2}$	$\pi(r_{1,2}) = 2$
r_4	$\pi(r_4) = 3$
r_1	$\pi(r_1) = 4$

Table 7.1: FMR-morphism: the ordered rule list

Chapter 8

Conflict Analysis Model

This chapter presents the supported policy anomalies and how they are identified. The anomaly analysis detection is computed on the Equivalent-Policy. Intra-policy and Inter-policy anomalies are based on previous works of Al-Shaer but have been extended to support a wider range of anomaly types. Inter-policy anomalies are categorized into three anomaly classes: *Blocked traffic anomalies*, *Encrypted traffic anomalies*, and *Modified traffic anomalies*. The blocked traffic anomalies include the Inter-policy filtering anomalies of Al-Shaer, and the encrypted traffic anomalies include the Inter-policy IPsec anomalies. This chapter concludes by presenting how the presented anomaly classes can be identified by relying on the Equivalent-Policy.

8.1 Intra-policy anomalies

The Intra-policy anomaly analysis is based on the works of Al-Shaer et al. [25] and Basile et al. [37]. Rule pair Intra-policy anomaly analysis can be applied to the Geometric-Model the same way as presented by Al-Shaer et al. [25]. However, by also taking into consideration the complete rule-set and the resolution strategy a more precise analysis can be performed. As already introduced by Basile et al. [37], the redundancy and the shadowing anomalies can be extended to the generally shadowing/redundancy anomaly. However, this generalisation can go a step further, and other classes can be introduced.

In this new classification, the correlation anomaly as introduced by Al-Shaer becomes obsolete because it is included in one of the extended anomaly types: generalization or shadowed. The new anomaly types are based on rule-sets and not rule-pairs. Therefore, correlated rules are always detected when evaluating them together with other intersecting rules or together with the default action.

8.1.1 Redundancy anomaly

The redundancy anomaly of Al-Shaer has been extend to verify also if a rule is redundant to a set of rules with equal action. In particular there are three cases: the rule covers a set of rules, the rule is covered by a set of rules, and the rule is partially covered and partially covers a set of rules.
A rule $r = (c, a)$ is completely unnecessary redundancy if the effective function returns an empty set and all rules covering the rule r have the same action a as the rule itself.

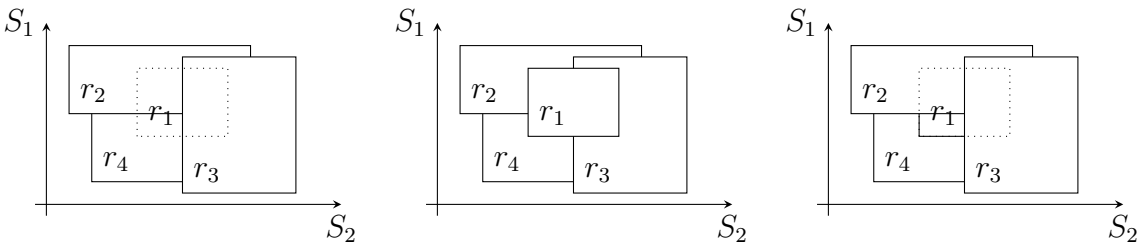
$$\text{eff}_p(r) = \emptyset \quad \wedge \quad \forall x \in c,\, p(x) = a$$

A rule $r = (c, a)$ is completely hidden redundant if the effective function returns the rule itself and the effective cover function returns an empty set.

$$\text{eff}_p(r) = r \quad \wedge \quad \text{effc}_p(r) = \emptyset$$

A rule $r = (c, a)$ is partially redundant if the effective function returns a non-empty set, all covering rules enforce the same action as the rule r and effective cover function returns an empty set.

$$\text{eff}_p(r) \neq \emptyset \neq r \quad \wedge \quad \text{effc}_p(r) = \emptyset \quad \wedge \quad \forall x \in c,\, p(x) = a$$



(a) Completely unnecessary redundancy (b) Completely hidden redundancy (c) Partially redundancy

Figure 8.1: Redundancy Anomaly types

Figure 8.1 shows examples for all three cases of redundancy. In all three examples the involved rules have the same condition clause and the same action. The only thing that changes is the priority between the rules. In Figure 8.1a rule r_1 is completely unnecessary redundant to the rules r_2, r_3 and r_4. In Figure 8.1b rule r_1 is completely hidden redundant by the rules r_2, r_3 and r_4. In Figure 8.1c rule r_1 is partially redundant, it covers rule r_4 and is covered by the rules r_2 and r_3.

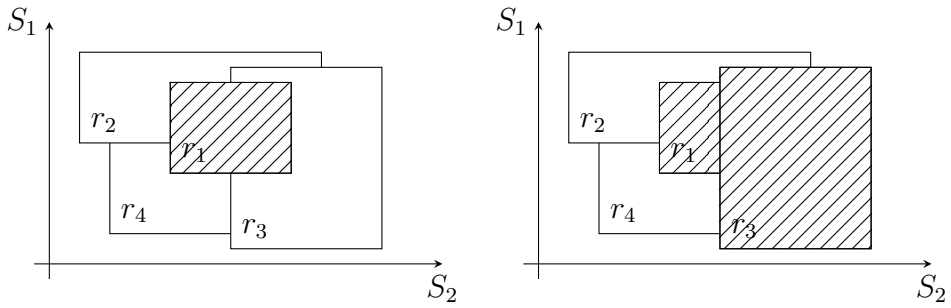
8.1.2 Generalization anomaly

The generalization anomaly of Al-Shaer can be extended by verifying if a rule is the generalization of a set of rules. Al-Shaer considers only rule pairs, where this new definition is still valid when the set contains only one rule.
A rule $r = (c, a)$ is a complete overriding generalization if the effective function returns an empty set and the effective cover function returns the rule itself.

$$\text{eff}_p(r) = \emptyset \quad \wedge \quad \text{effc}_p(r) \neq \emptyset = r \quad \wedge \quad \forall x \in c,\, p(x) = a$$

A rule $r = (c, a)$ is a partially generalization if the effective function returns the same non-empty set as the effective cover function and all covering rules enforce the same action as the rule r itself.

$$\text{eff}_p(r) = \text{effc}_p(r) \neq \emptyset \quad \wedge \quad \forall x \in c,\, p(x) = a$$



(a) Complete overriding generalization

(b) Partially generalization

Figure 8.2: Generalization Anomaly types

Figure8.2 shows the two cases of the generalization anomalies. Figure 8.2a shows an example of the complete generalization anomaly, where rule r_1 is the generalization of the set of rules r_2, r_3 and r_4. Figure 8.2b shows an example of the generally generalization anomaly, where rule r_1 is the generalization of the set of rules r_2 and r_3.

This extended definition of the generalization anomaly is one of the new anomaly types that made the correlated rule anomaly obsolete. This new definition of the generalization anomaly, in general, is based on correlated rules that are evaluated contemporaneously.

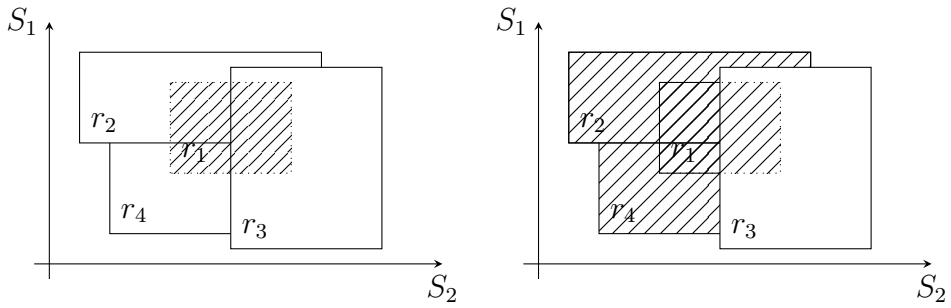
8.1.3 Shadowing anomaly

The shadowing anomaly of Al-Shaer can be extended by verifying if a rule is shadowing a set of rules. Al-Shaer considers only rule pairs, where this new definition is still valid when the set contains only one rule.
A rule $r = (c, a)$ is completely shadowed if the effective function returns an empty set and all covering rules enforce a different action as the rule r itself.

$$\text{eff}_p(r) = \emptyset \quad \wedge \quad \forall x \in c,\, p(x) \neq a$$

A rule $r = (c, a)$ is partially shadowed if the effective function returns a non-empty set, the effective cover function returns an empty set and all covering rules enforce a different action as the rule r itself.

$$\text{eff}_p(r) \neq \emptyset \quad \wedge \quad \text{effc}_p(r) = \emptyset \quad \wedge \quad \forall x \in c \setminus \text{eff}_p(r),\, p(x) \neq a$$



(a) Completely shadowing (b) Partially shadowing

Figure 8.3: Shadowing anomaly types

Figure 8.3 shows the two cases of the shadowing anomalies. Figure 8.3a shows an example of the completely shadowing anomaly, where rule r_1 is shadowed by the set of rules r_2, r_3 and r_4. Figure 8.3b shows an example of the partially shadowing anomaly, where rule r_1 is shadowed by the rule r_3.

This extended definition of the shadowing anomaly is the second new anomaly type that made the correlated rule anomaly obsolete. This new definition of the shadowing anomaly, in general, is based on correlated rules that are evaluated contemporaneously. Another instance of correlated rules that is included is when the default action is equal to the action enforced by the lower-priority rule. In this case, a partially shadowing of two rules is detected.

8.1.4 Generally hidden

The generally hidden anomaly is a complete new anomaly class. It occurs when a rule is completely covered by a set of rules enforcing different actions.
A rule $r = (c, a)$ is generally hidden if the effective function returns an empty set, some covering rules enforce a different action as the rule r itself and some covering rules enforce the same action as the rule r itself.

$$\text{eff}_p(r) = \emptyset \quad \wedge \quad \exists x \in c,\, p(x) \neq a \quad \wedge \quad \exists x \in c,\, p(x) = a$$

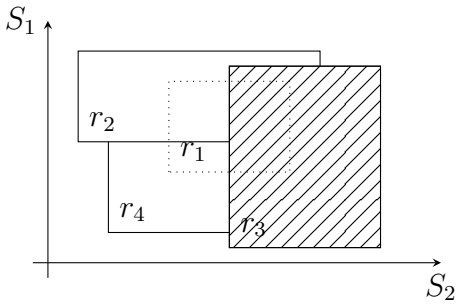


Figure 8.4: Generally Hidden

Figure 8.4 shows an example of this anomaly. The rule r_1 is completely covered by the rules r_2, r_3 and r_4.

8.2 Inter-policy anomalies

Inter-policy anomalies occur among rules of different policies, this includes also policies of different type. As already proposed for Intra-policy anomalies, this work also extends the definition of Al-Shear at al. [26] for Inter-policy anomalies. It is not sufficient to evaluate rule pairs to correctly identify all possible anomalies. Therefore, the proposed model consider the enforced actions of the involved policies on a packeted basis.

The filtering anomalies of Al-Shaer are represented by the *Blocked traffic anomalies*. This anomaly class does not only consider filtering actions such as ALLOW and DENY, but includes all possible actions.

The IPsec anomalies of Al-Shaer are represented by the *Transformed traffic anomalies*. This anomaly class also contains two new anomalies: the *Encrypted traffic anomalies* and the *Modified traffic anomalies*

8.2.1 Blocked traffic anomalies

The blocked traffic anomaly may be caused by a filtering policy or by a routing policy. The first case is when a security policy contains a rule for a packet that has been blocked by an upstream filtering policy. The second case is when a security policy contains a rule for a packet that has not been forwarded by an upstream routing policy.

The proposed algorithm is based on the rule relations present in the Equivalent-Policy. Therefore, some anomalies that are detected by the model of Al-Shaer are ignored by this model because they are hidden by other rules and, therefore, not influence the enforced action. Furthermore, the correlation anomaly as defined by Al-Shaer is going to be substituted by more precise definition. Rules that are classified as correlated by Al-Shaer are classified by this model as shadowing or spuriousness depending on the enforced action of the rules. This is because only the effective part of the rules is taken into consideration and, therefore, the correlation definition is not applicable.

Shadowing Anomaly

The shadowing anomaly occurs when a downstream security policy blocks a packet that has been allowed by an upstream security policies. Formally, two policies p_a and p_b contain a shadowing anomaly if there are two rules $r_i \in p_a$ and $r_j \in p_b$ with the following properties:

$$X = \text{eff}_{p_a}(r_i) \cap \text{eff}_{p_b}(r_j) \neq \emptyset \quad \wedge \quad \forall x \in X \;\; p_a(x) = \text{ALLOW} \quad \wedge \quad p_b(x) = \text{DENY}$$

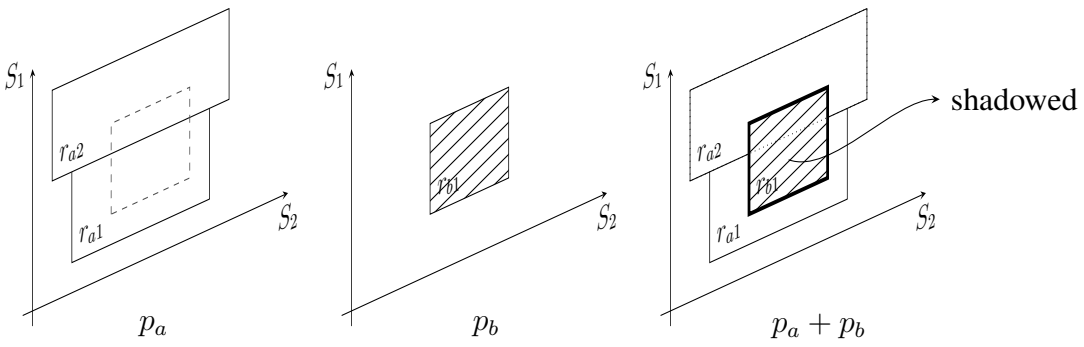


Figure 8.5: Inter-policy Shadowing Anomaly

Figure 8.5 shows an example of a shadowing anomaly between two cascaded security policies. The first policy p_1 contains two rules r_{a1} and r_{a2}, both enforcing the ALLOW action and the second policy p_2 contains one rule r_{b1} enforcing the DENY action. The composition of these two policies results in a new policy $p_1 + p_2$ where rule r_{b1} is shadowing one part of rule r_{a1} and one part of rule r_{a2}.

Spuriousness Anomaly

The spuriousness anomaly occurs when a downstream security policy allows a packet that has already been blocked by an upstream security policy. Formally, two policies p_a and p_b contain a spuriousness anomaly if there are two rules $r_i \in p_a$ and $r_j \in p_a = b$ with the following properties:

$$X = \text{eff}_{p_a}(r_i) \cap \text{eff}_{p_b}(r_j) \neq \emptyset \quad \wedge \quad \forall x \in X \;\; p_a(x) = \text{DENY} \quad \wedge \quad p_b(x) = \text{ALLOW}$$

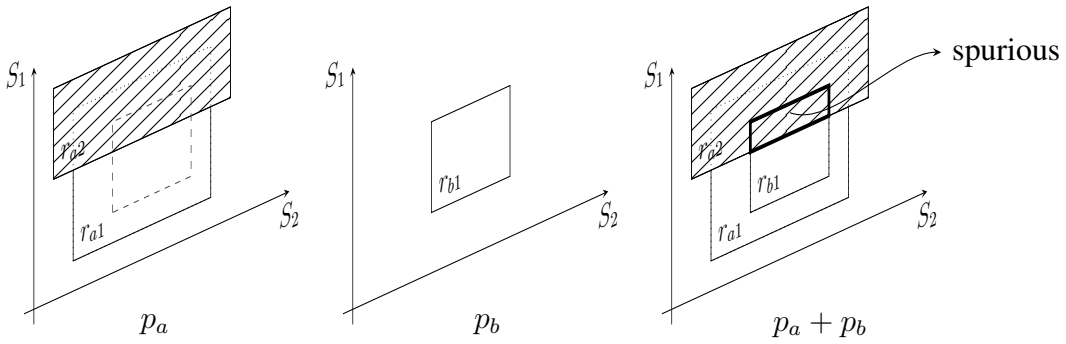


Figure 8.6: Inter-policy Spuriousness Anomaly

Figure 8.6 shows an example of a spuriousness anomaly between two cascaded security policies. The first policy p_1 contains two rules, one r_{a1} that enforces the ALLOW action and one r_{a2} that enforces the DENY action. The second policy p_2 contains one rule r_{b1} enforcing the ALLOW action. The composition of these two policies results in a new policy $p_1 + p_2$ where rule r_{b1} is spurious on the one part of rule r_{a2} but has no anomaly to the other part of rule r_{a1}.

Redundancy Anomaly

The shadowing anomaly occurs when a downstream security policy blocks a packet that is also blocked by an upstream security policy. Formally, two policies p_a and p_b contain a redundancy anomaly if there are two rules $r_i \in p_a$ and $r_j \in p_b$ with the following properties:

$$X = \text{eff}_{p_a}(r_i) \cap \text{eff}_{p_b}(r_j) \neq \emptyset \quad \wedge \quad \forall x \in X \;\; p_a(x) = \text{DENY} \quad \wedge \quad p_b(x) = \text{DENY}$$

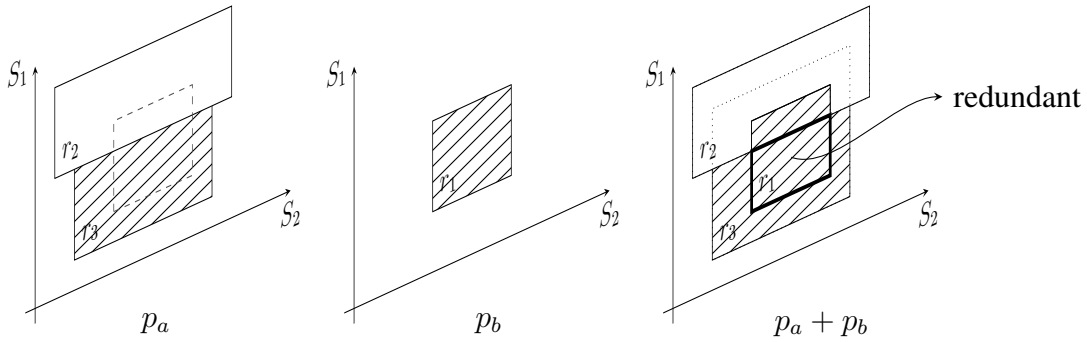


Figure 8.7: Inter-policy Redundancy Anomaly

Figure 8.7 shows an example of a redundancy anomaly between two cascaded security policies. The first policy p_1 contains two rules, one r_{a1} that enforces action DENY and one r_{a2} that enforces action ALLOW. The second policy p_2 contains one rule r_{b1} enforcing action DENY. The composition of these two policies results in a new policy $p_1 + p_2$ where rule r_{b1} is redundant on the one part of rule r_{a1} but has no anomaly to the other part on rule r_{a2}.

8.2.2 Transformed traffic anomalies

Transformed traffic anomalies occur when a security policies applies a transformation to a rule, therefore, cannot be matched by a downstream policy. The transformed traffic anomalies are: the *Encrypted traffic anomaly*, the *Overlapping session anomaly* and the *Modified traffic anomaly*.

Encrypted traffic anomaly

The encrypted traffic anomaly occurs when a security policy encrypts a packet for which another security policy has defined a rule. In this case, the packet can not be matched

because the required packet fields are encrypted and, therefore, the rule will never be applied.

Formally two policies p_a and p_b contain an encrypted traffic anomaly if there are two rules $r_i \in p_a$ and $r_j \in p_b$ with the following properties:

$$X = \text{eff}_{p_a}(r_i) \cap \text{eff}_{p_b}(r_j) \neq \emptyset \ \wedge$$

$$\forall x \in X \ \ p_a(x) = \text{ENCRYPT} = x_0 \ \wedge$$

$$x_0 \notin \text{eff}_{p_b}(r_j)$$

Overlapping session anomaly

The overlapping session anomaly occurs when a security policy encrypts a packet that is also encrypted by another security policy. In this case, however, the packet is correctly matched by the second policy.

Formally two policies p_a and p_b contain an overlapping session anomaly if there are two rules $r_i \in p_a$ and $r_j \in p_b$ with the following properties:

$$X = \text{eff}_{p_a}(r_i) \cap \text{eff}_{p_b}(r_j) \neq \emptyset \ \wedge$$

$$\forall x \in X \ \ p_a(x) = \text{ENCRYPT} = x_0 \ \wedge$$

$$p_b(x_0) = \text{ENCRYPT} = x_0'$$

Modified traffic anomaly

The modified traffic anomaly occurs when a security policy modifies the packet header and/or the payload and, therefore, a downstream policy can not match it any more. This anomaly originates form security policies that enforce the MODIFY or the NEW HEADER action.

Formally two policies p_a and p_b contains a modified traffic anomaly if there are two rules $r_i \in p_a$ and $r_j \in p_b$ with the following properties:

$$X = \text{eff}_{p_a}(r_i) \cap \text{eff}_{p_b}(r_j) \neq \emptyset \ \wedge$$

$$\forall x \in X \ \ p_a(x) = \text{MODIFY} = x_0 \ \wedge$$

$$x_0 \notin \text{eff}_{p_b}(r_j)$$

8.3 Anomaly verification

The previous sections have presented all possible anomaly types within a single security policy or between security policies of different type. The identification of these anomalies relies on the effective function eff_p computation and the effective cover function effc_p. These two functions are complex to compute because they require the computation of set union operation, however the Equivalent-Policy supports them. Indeed, the evaluation of these two functions is very efficient when using key properties of the canonical form and the semi-lattice. The semi-lattice, in particular its cover graph representation, enables the computation of this two functions without resorting to the set union.

8.3.1 Effective function computation

The effective function $\mathrm{eff}_p(r)$ can be computed by evaluating the outgoing vertices $r' = (c', a')$ of the rule r in the cover graph $G(p)$:

$$\mathrm{eff}_p(r) = \emptyset \iff \bigcup_{r \prec r'} c' \cap c = c$$

The condition verification for this function is performed in two steps, first the simple verification and then the full verification.

The first step verifies that each selector of the rule r is a subset of the union of each selector in the covering rules. Should this verification fail, the algorithm concludes that $\mathrm{eff}_p(r) \neq \emptyset$, otherwise the next step is performed.

The second step verifies that the union of all intersections between the condition clause of the rule r and the covering condition clauses is equal to the condition clause of the rule r itself. Should this verification fail, the algorithm concludes that $\mathrm{eff}_p(r) \neq \emptyset$, otherwise $\mathrm{eff}_p(r) = \emptyset$.

8.3.2 Effective cover function computation

The effective cover function $\mathrm{effc}_p(r)$ can also be computed by evaluating the outgoing vertices $r' = (c', a')$ of the rule r in the cover graph $G(p)$:

$$\mathrm{effc}_p(r) = \emptyset \iff \bigcup_{r \prec r'} c' \cap c = c$$

The condition verification for this function is performed by evaluating if all nodes in the semi-lattice of the rule r, where the rule is not covered, enforce the same action a also after removing the rule r. Should this verification fail the algorithm concludes that $\mathrm{eff}_p(r) \neq \emptyset$, otherwise $\mathrm{eff}_p(r) = \emptyset$.

Chapter 9

Reachability Analysis Model

This chapter presents the algorithms and data models required to perform an offline reachability analysis based on the Equivalent-Policy. The proposed reachability analysis is based on a custom query structure, which is also presented in this chapter. The query structure allows the definition of different reachability scopes. The reachability model evaluates the query and returns the reachability results.

9.1 The model

Offline reachability analysis is based on reachability queries. Reachability queries define the reachability condition that must be verified. The reachability model takes in input the query, evaluated, and the result is returned. Figure 9.1 shows this basic concept.

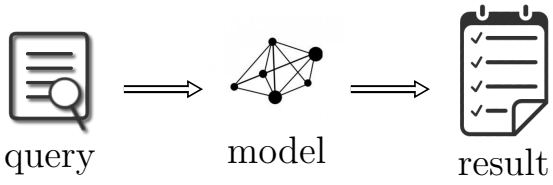


Figure 9.1: Offline reachability model

Reachability queries can define a very precise reachability condition such as: Can host A reach the Web-server on host B? However, they can also express much more complex reachability conditions, including multiple host, entire subnets, connection states, and routing information. The complexity of a reachability analysis model relies on the expressiveness of the reachability query language how is is evaluated. This section presents

the basic concepts that are included in the proposed reachability analysis model. Furthermore, the key reachability conditions are presented and how the model evaluates them.

All reachability queries are executed on the precomputed Equivalent-Policy, not on the network model itself. This has the advantage that the number of involved policy rules is drastically reduced, and the internal complexity of the network can be ignored. The Equivalent-Policy requires fewer rules to describe the end-to-end security behaviour of a computer network because it eliminates all redundant rules. Furthermore, it is possible to identify the cause of a reachability problem since rules in the Equivalent-Policy maintain a reference to the original rules. The computation of the Equivalent-Policy is described in previous chapters and, therefore, will not be presented here.

An Equivalent-Policy is always defined between two zones in the computer network. In case a reachability query involves more than one source-zone and/or more than one destination zone, a *multi-zone Equivalent-Policy* is created on the fly. Since different zones have non-overlapping IP addresses, also rules from different Equivalent-Policy are not correlated. The creation of the multi-zone Equivalent-Policy has no impact on the performance since it is merely the union of two or more Equivalent-Policies.

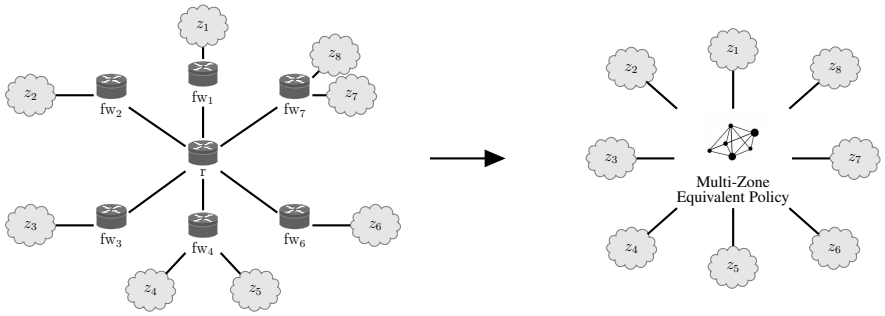


Figure 9.2: Multi-zone Equivalent-Policy

The routing policy defines the path in a computer network that a packet must follow. However, the default path may change in case of routing changes and, therefore, these alternative paths are also part of the analysis. The reachability algorithm supports two types of analysis: static analysis for the default path and dynamic analysis for alternative paths. The dynamic reachability analysis takes into account all possible paths from source to destination. The type of analysis to perform is specified by the reachability query.

The dynamic analysis has the objective to identify different reachability properties on the alternative paths. The reachability analysis handles differences among paths according to a defined strategy: *lower bound*, *upper bound* and *highlight path anomalies*. The lower bound strategy evaluates a packet as allowed only if all it is allowed on all possible paths.

The upper bound strategy evaluates a packet as allowed when it is allowed at least on one path. The highlight path anomalies strategy highlights all possible packets which are treated differently on different paths and evaluates a packet as allowed or denied only if all it is allowed or denied on all possible paths.

The algorithm is also able to support different connection types in addition to the static and dynamic reachability analysis. The supported connection oriented queries are: *stateless communication*, *stateful communication*, and *multiple stateful communication*. The stateless communication query evaluates if a packet send by the source can reach the destination. The stateful communication query evaluates if a packet send by the source can reach the destination and that the destination can send a reply. The multiple stateful communication query evaluates if a host can establish more than one stateful connection to a destination.

The algorithm transforms the reachability query into a rule condition, the so-called query condition. The query condition, just like a rule condition, is a set of selectors s_i representing the values of a packet field. The values of the packet fields are defined in the reachability query. The query condition is applied to the Equivalent-Policy by means of the *query matching function* $match_q()$. This function identifies all policy rules that are relevant for the query evaluation. Depending on the analysis type (static, dynamic) the query condition is applied to one or multiple Equivalent-Policies. The resulting rule sets are the base for the query result.

9.2 Reachability Queries

The reachability query allows the user to specify the details about the reachability analysis he wants to perform. In literature have been different proposals for reachability query languages. One of the first languages proposes was SFQL, the Structured Firewall Query Language [97]. SFQL was designed to query one single firewall and has the following syntax:

$$\begin{aligned}&\texttt{select } S_i\\&\texttt{from } f\\&\texttt{where } (X_1 \in S_1) \wedge \cdots \wedge (X_k \in S_k) \wedge (\text{action} = \langle \text{dec} \rangle)\end{aligned}$$

S_i represents the protocol field (e.g., source or destination address) that must be present in the result. X_i are non-empty subsets of protocol fields S_i and are used to restrict the query domain. $\langle \text{dec} \rangle$ specifies if the query searches for allowed or denied communications.

The main limitation of SFQL is that it supports only one protocol field in the `select` clause. Therefore, the expressiveness of the query result is very limited. To overcome the limitations of SFQL Liu et al. extended the language and defined the Structured Reachability Query Language (SRQL) [75]. SRQL is designed to support multiple firewalls and has the following syntax:

`reachability_type` $\mathcal{T}$
`connection_type` $\mathcal{O}$
`select` S_i
`where` $(X_1 \in S_1) \wedge \cdots \wedge (X_k \in S_k) \wedge (\text{action} = \langle \text{dec} \rangle)$

SRQL supports three reachability types $\mathcal{T}$ = instantaneous, $\mathcal{T}$ = upper-bound and $\mathcal{T}$ = lower-bound. The instantaneous reachability is calculated over a pre-defied path, whereas the other two take into consideration all possible paths. The upper-bound reachability is the union of the allowed communications of all paths. Therefore, a communication is evaluated as allowed by the reachability model when all paths allow the communication. The lower-bound reachability is the intersection of the allowed communications of all paths. Therefore, a communication is evaluated as allowed by the reachability model when at least one paths allows the communication.

SRQL also supports two connection type: $\mathcal{O}$ =connection-oriented and $\mathcal{O}$ =connectionless. Connection-oriented protocols is, for example, the Transmission Control Protocol (TCP). Connectionless protocols are for example IP and User Datagram Protocol (UDP).

S_i represents the protocol field that must be present in the result: source and destination IP address (S, D), source and destination port (SP, DP), and protocol type (PT. X_i are non-empty subsets of protocol fields S_i and are used to restrict the query domain.

9.2.1 Query format

The query language used in the proposed model is based on SRQL but has more features significantly. The new query language has the following syntax:

`reachability_type` $\mathcal{T}$
`connection_type` $\mathcal{O}$
`fixed_path` $\mathcal{P}$
`Equivalent-Policy` $\mathcal{E}$
`select` $\mathcal{F}$
`where` $\mathcal{B} \wedge (\text{action} = \langle \text{dec} \rangle)$

$\mathcal{T}$ defines the type of reachability to perform, allowed reachability types are:

- instantaneous queries (I)
- lower-bound (LB)
- upper-bound (UB)
- unspecified strategy (U)
- static routing information (S)

$\mathcal{O}$ defines the connection type to use:

- stateful (SF)
- stateless (SL)
- multiple stateful (MSF)

$\mathcal{P}$ defines the path from source to destination expressed as a list of network nodes

$\mathcal{E}$ defines when the Equivalent-Policy is calculated:

- zone-to-zone Equivalent-Policy (Z)
- query-specific Equivalent-Policy (Q)

$\mathcal{F}$ defines the set of packet fields S_i used in the query:

- source (*S*) and destination (*D*) IP address
- source (*SP*) and destination (*DP*) port
- protocol type (*PT*)
- protocol state (*PS*)
- application protocol (*AP*)
- time (*T*) and date (*D*)
- HTTP_method (*QM*), HTTP_req_header (*QH*), HTTP_rep_header (*PH*)
- hit-limit (*L*)
- hit-limit-burst (*LB*)
- URL domains (*UD*), URL paths (*UP*), URLs (*U*)

$\mathcal{B}$ defines a Boolean expression of conditions on the previous fields:
$(F_1 \in S_1) \wedge \cdots \wedge (F_k \in S_k) \wedge (\text{action} = \langle \text{dec} \rangle)$

9.2.2 Query condition

The query condition is created based on the Boolean expression $\mathcal{B}$ from the reachability query. Since the boolean expression is defined over a set of packet fields the transformation from a reachability query to the query condition is straightforward. Each packet field in the boolean condition corresponds to a field in the query condition.

For example, the query condition representing the following reachability query

```
reachability_type LB
connection_type SL
select S, SP, D, DP, PT, PS
where D = 10.1.1.8   ∧ DP = 80   ∧ action = ALLOW
```

is $c_q = s_1 \times s_2$ with $s_1 = 10.1.1.8$ and $s_2 = 80$ and its selection space $\mathfrak{S} = \mathbb{F}_1 \times \mathbb{F}_2$ is defined over $\mathbb{F}_1$ for the destination address and $\mathbb{F}_2$ for the destination port.

9.2.3 Query matching function

The query matching function Qmatch_R returns the subset Q of rules $r_i = (c_i, a_i)\, r_i \in R$ whose condition $c_i = s_1 \times s_2 \times \cdots \times s_m$ intersects the query condition c_q.

$$\begin{aligned} \mathrm{Qmatch}_R : \quad \mathfrak{S} &\longrightarrow 2^R \\ x &\longmapsto M = \{r_i \in R \mid c_q \cup c_i \neq \emptyset\} \end{aligned}$$

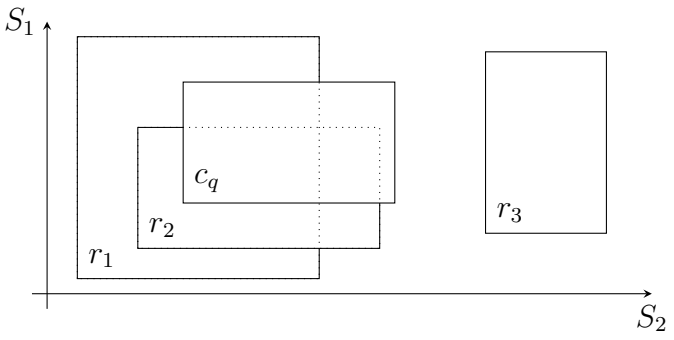


Figure 9.3: Example of Qmatch_R

Figure 9.3 shows an example of the query matching function Qmatch_R applied to the rule set $R = \{r_1, r_2, r_3\}$ for query condition c_q. The query condition c_q is intersecting the rules r_1 and r_2. Therefore, the query matching function returns a set containing these two rules $\mathrm{Qmatch}_R(c_q) = \{r_1, r_2\}$. Rule r_3 is not intersecting the query condition c_q and, therefore, it is not contained in the result set.

9.3 Query result

The result of a reachability analysis is composed of four fields: (answer, query result domain, stateful domain and accuracy):

- *answer*: The answer field is signed to *Allow*, *Deny* or *Partly*. It is Allow if the traffic defined in the reachability query is always allowed. It is Deny if the traffic defined in the reachability query is always blocked. It is Partly if only a subset of the traffic defined in the reachability query is allowed.
- *query result domain*: The query result domain represents all the allowed traffic, it is expressed as a set of rules derived from the Equivalent-Policy. This field is only used when the answer is *Partly*, and it is restricted according to the query conditions (via intersection).
- *stateful result domain*: The stateful domain field represents all the communications from the destination to the source that are allowed by some stateful rules as replies to allowed communications. This field is only used for stateful queries, and the answer is *Partly*.
- *accuracy*: The accuracy field highlights possible irregularities in the result. Irregularities may occur when the path contains unsupported security policies or when any of the involved security policies contains unsupported selectors.

The following subsections present the result domain and the query accuracy in more detail. Some practical examples are included for better understanding. The result domain as explained in the next subsection is valid for both the query result domain and the stateful result domain.

9.3.1 Result domain

By default both, the query result domain and the stateful result domain, include all selectors and associated values in order to provide all information. The user can specify in the `select` clause which selectors to include in them if only a subset is required. However since the query result domain and the stateful result domain is modelled just like a normal policy, the user can also navigate (e.g., to inspect fields where conditions are non-tautological) them directly.

The query asks if a host in the subnet 10.1.2.0/24 is allowed to establish a TCP connection with a service 10.1.1.8:80. The following query is an example which is performed on the policy in Table 9.1:

```
reachability_type LB
connection_type SL
select S, SP, D, DP, PT, PS
where
S = 10.1.2.0/24  ∧
D = 10.1.1.8  ∧
DP = 80  ∧
PT = TCP  ∧
action = ALLOW
```

	Source IP	S.Port	Dest IP	D.Port	Proto	Protocol States	Action
r_1	any	any	any	any	any	ESTABLISHED	ALLOW
r_2	10.1.2.7	any	10.1.1.9	any	any	any	DENY
r_3	10.1.2.7	any	10.1.1.19	any	any	any	DENY
r_4	10.1.2.0/24	>1023	10.1.1.0/24	80	TCP	NEW	ALLOW
r_5	10.2.2.1	>1023	10.1.1.1	80	TCP	NEW	ALLOW
...							
	*	*	*	*	*	*	DENY

Table 9.1: Example stateful filtering policy.

In this case, the hosts are only allowed to establish a TCP connection using a source port greater than 1023. Therefore, the answer field is set to *Partly*, and the query result domain contains the following expression:

$$\boldsymbol{S} \in 10.1.2.0/24 \wedge \boldsymbol{SP} > 1023 \wedge \boldsymbol{D} = 10.1.1.8 \wedge$$
$$\wedge \boldsymbol{DP} = 80 \wedge \boldsymbol{PT} = \boldsymbol{TCP} \wedge \boldsymbol{PS} = any$$

where *PS* indicates conditions on protocol states.

In some cases, the query result domain can also be represented *compactly*. For example, the previous query result domain can also be represented as $\mathrm{SP} > 1023$. The compact representation is possible if and only if the allowed and blocked connections can be expressed as a single Boolean expression containing only ANDs ($(X_1 \in S_1) \wedge \cdots \wedge (X_k \in S_k)$).

However, this is not always possible and therefore, in this cases, the query result domain is best expressed as security policy. An example of such a query:

```
reachability_type LB
connection_type SL
select S, SP, D, DP, PT, PS
where
S = 10.1.2.0/24  ∧
D = 10.1.1.0/24  ∧
action = ALLOW
```

This query asks if there is a host in the subnet 10.1.2.0/24 which can reach any service in the subnet 10.1.1.0/24. The result domain of this query is as follows:

$$(\boldsymbol{S} = 10.1.2.7 \wedge \boldsymbol{SP} = * \wedge (\boldsymbol{D} = 10.1.1.9 \vee \boldsymbol{D} = 10.1.1.19) \wedge$$
$$\boldsymbol{DP} = 80 \wedge \boldsymbol{PT} = \boldsymbol{TCP}) \quad \wedge \quad \neg(\boldsymbol{S} \in 10.1.2.0/24 \wedge$$
$$\boldsymbol{SP} = * \wedge \boldsymbol{D} \in 10.1.1.0/24 \wedge \boldsymbol{DP} = * \wedge \boldsymbol{PT} = *)$$

Representing the result domain as a security policy is much clearer as Table 9.2 shows.

Source IP	S.Port	Destination IP	D.Port	Proto	Protocol States	Action
10.1.2.7	any	10.1.1.9 OR 10.1.1.19	any	any	any	DENY
10.1.2.0/24	>1023	10.1.1.0/24	80	TCP	NEW	ALLOW
*	*	*	*	*	*	DENY

Table 9.2: Query result in policy format.

9.3.2 Query result accuracy

The query result accuracy has been ignored in previous research, although it is crucial. The accuracy is influenced by unsupported security policies or unsupported rule conditions within the analysed computer network. Formally this means that the query result domain is not tautological in the domain of the unsupported condition.

For example, the path from source to destination contains an application level firewall which blocks all encrypted traffic, a model which does not support application level firewalls will incorrectly conclude that the traffic is permitted. In this case, the accuracy field will flag the reachability result as potentially false. This has the advantage that although the model does not support a certain security policy, the analysis can still be performed, and the user is informed about possible inconsistencies.

Chapter 10

Policy Comparison Model

This chapter presents the proposed policy comparison model based on the Equivalent-Policy. In particular, the four different comparison scopes (Single Policy Change-impact-analysis, Single Policy Implementation-verification, Multiple Policy Change-impact-analysis and Multiple Policy Implementation-verification) are explained.

10.1 Model

Policy comparison is no easy task because the policies with a very different rule structure may enforce the same action. The following example shows such a case: the two policies p_1 and p_2 respectively use both the FMR resolution strategy and have the default action DENY. They are composed of entirely different rules but enforce the same action to all packets. Both policies will block packets with destination address 10.1.5.2 and destination port 80 and 81. Policy p_1 contains the following rules:

$$\begin{aligned} r_1 = &\quad (D = 10.1.5.2, \quad DP = 80{,}81) \quad \rightarrow \text{DENY} \\ r_2 = &\quad (D = 10.1.5.2, \quad DP = *) \quad\quad\quad \rightarrow \text{ALLOW} \end{aligned}$$

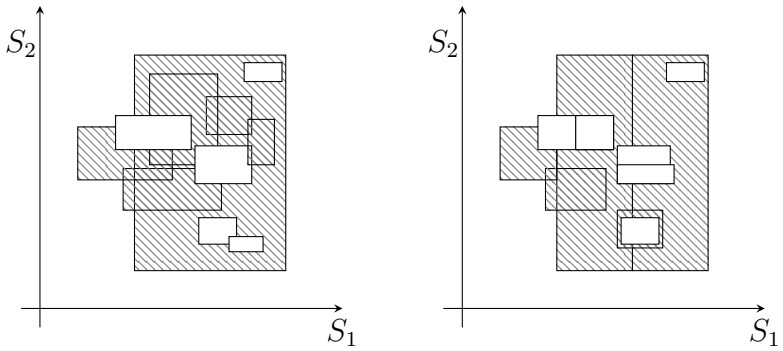
Policy p_2 contains the following rules:

$$\begin{aligned} r_1 = &\quad (D = 10.1.5.2, \quad DP = 1 - 79) \quad\quad \rightarrow \text{ALLOW} \\ r_2 = &\quad (D = 10.1.5.2, \quad DP = 82 - 1024) \quad \rightarrow \text{ALLOW} \\ r_3 = &\quad (D = 10.1.5.2, \quad DP > 1024) \quad\quad\quad \rightarrow \text{ALLOW} \end{aligned}$$

Policy p_1 defines a rule with higher propriety that blocks specifically the undesired ports and a second low priority rule that permits the traffic on all ports. Therefore, a packet with destination port 80 or 81 is matched by the first rule and blocked. Policy p_2 defines

only rules with ALLOW action and packets that should be blocked are enforced by the default action. Therefore, a packet with destination port 80 or 81 is matched by no rule and blocked by the default action.

A more complex example of such a case is shown in Figure 10.1. In this case, the two policies (Policy 1 and Policy 2) have much more rules and the rule structure of the two policies is very different. The enforced action of both policies, however, is the same in the whole decision space. The comparison of the two policies is quite easy by having the geometric representation side-by-side. Therefore, can policies be effectively compared by evaluating their geometric superposition.



(a) Policy 1 (b) Policy 2

Figure 10.1: Policy comparison example

The Geometric-Model is, therefore, best suited for this task. However, the representation of policy differences is no easy task. Differences can have a very complex structure and are difficult to represent in an easily readable way. Without proper representation, however, the computed differences are without meaning.

The comparison among the two policies, policy p_a and policy p_b, is a good example. Both policies use the FMR resolution strategy and have the default action DENY. Policy p_a permits traffic for the destination address within the subnet 10.0.0.0/16 and directed to the destination port between 80 and 100. Policy p_b permits traffic for the destination address within the subnet 10.0.0.0/16 and directed to the destination port between 90 and 110.

Policy p_a composed by one rule:

$$r_{a1} = (D = 10.0.0.0/16, DP = 80 - 100) \rightarrow \text{ALLOW}$$

Policy p_b is also composed by one rule:

$$r_{b1} = (D = 10.0.0.0/16, DP = 90 - 110) \rightarrow \text{ALLOW}$$

The differences between the two policies p_a and p_b are well highlighted by superposing the two policies in the decision space. Figure 10.2 highlights the differences between the two policies in grey.

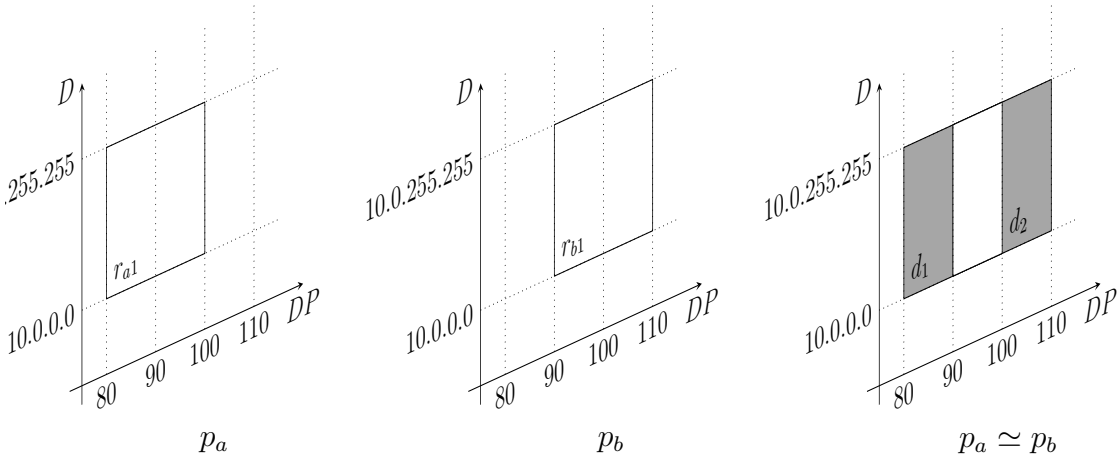


Figure 10.2: Comparison example between policy p_a and p_b

The differences between these two policies can also be represented as two hyper-rectangles in the decision space:

$$d_1 = (D = 10.0.0.0/16, DP = 80 - 90)$$

$$d_2 = (D = 10.0.0.0/16, DP = 100 - 110)$$

However, in general, differences can not be represented with a single condition clause (that is, one single compact hyper-rectangle).

The relative complement of two n-dimensional hyper-rectangles, requires up to $n - 1$ hyper-rectangles to be represented. The relative complement representation of more than two hyper-rectangles requires exponentially more. However this is not a limitation of the Geometric-Model itself, but it is also valid for the all other representations. However, other approaches are also no valid solution, for example decorrelating the rules (Liu et al. [84]). Because also in this approaches the calculation of the relative complement is required and the number of resulting rules would be exponential. Therefore, is the relative complement representation not suitable for representing rule differences based on this considerations.

Another important aspect to consider is that one and the same difference can be represented in different ways. The proposed model solves this issues by representing the result

of the comparison as a policy, called *Difference Policy*. The policy itself is represented by the Geometric-Model as a semi-lattice where the rule actions are substituted by the two fictitious actions, 'identical' (ID) and 'different' (DIFF). The *Difference Policy* can formally be represented as:

$$d = (D, \mathfrak{R}, E, \mathrm{ID})$$

For instance, the differences between p_a and p_b could be represented by a policy with default action ID composed of these three rules:

$$d_1 = (D = 10.0.0.0/16, D = 80 - 100) \rightarrow \mathrm{DIFF}$$

$$d_2 = (D = 10.0.0.0/16, D = 90 - 110) \rightarrow \mathrm{DIFF}$$

$$d_{1,2} = (D = 10.0.0.0/16, D = 90 - 100) \rightarrow \mathrm{ID}$$

The resulting Difference Policy d is a very compact representation of the differences. Its rule set D contains the minimal number of rules and, in case, the compared policies are equal, it contains no rule at all.

Another advantage of this approach is that network administrators are already familiar with the policy format. Therefore differences represented as a policy are easy to comprehend for administrators. Furthermore, the rules in the Difference Policy d have a direct correlation to the original rules and, therefore, it is very clear which rules are the cause of the differences.

10.2 Algorithm

This section presents the algorithm applied in all four policy comparison scopes. The algorithm creates the Difference Policy, as presented in the previous section, and revives as input two policies p_a and p_b. Before going in detail, for better understanding a simplified version of the algorithm is presented:

- Each rule in policy p_a that enforces a different action as the default action of policy p_b is added to D.
  - With respect to the previous example, r_{a1} satisfies this condition and, therefore, d_1 is added in D
- Each rule in policy p_b that enforces a different action as the default action of policy p_a is added to D.

  - With respect to the previous example, r_{b1} satisfies this condition and, therefore, d_2 is added in D

- For each rule in D that intersects another rule and have the same action, a new rule is created that has as condition clause the intersection of the condition clauses of the two rules and as action ID.

  - With respect to the previous example, d_1 and d_2 intersect each other and therefore $d_{1,2}$ (that has $c_1 \cap c_2$ as condition clause and ID as action) is added to D.

- All rules created by intersections of other rules have higher priority than the original rules.

  - With respect to the previous example, d_1 and d_2 have a lower priority then $d_{1,2}$.

- All rules in D with action ID that do not intersect any rule with action DIFF are removed. Indeed, they enforce the same action as the default action. Therefore, they do not add any information.

From a computational point of view, this simplified algorithm is very inefficient. However, by using the canonical form generation algorithm (Section 7.2) and the Semantic preserving morphism (Section 7.3) a much more efficient algorithm can be created.

To use the canonical form generation algorithm, some parts must change. The rule composition (Section 7.2) remains the same as presented. The policy composition (Section 7.2) must be modified for the integration.

The rule composition "$\circ$" of k rules $r_i \in R$ in a policy p is formally defined as:

$$\begin{array}{rccc} \circ: & R \times R \times \cdots \times R & \rightarrow & k^{\mathfrak{S}} \times \mathcal{A} \\ & r_{1,2,\cdots,k} = r_1 \circ r_2 \circ \cdots \circ r_k & \mapsto & (c_1 \cap c_2 \cap \cdots \cap c_k, \mathfrak{R}(\{r_1, r_2, \ldots, r_k\})) \end{array}$$

The policy composition computes when applied to the Equivalent-Policy, the resulting action depending on where the policy is allocated (in serial or parallel). In this case, however, the important aspect to evaluate is not which action prevails over another, but if the actions are equal or different. The *comparison composition* function models these interactions between policies:

$$\sim: \mathcal{A} \times \mathcal{A} \rightarrow \mathcal{A}$$

it describes the comparison of actions of two security policies:

$$\begin{aligned} a \sim a = d \sim d &= \text{ID} \\ a \sim d = d \sim a &= \text{DIFF} \end{aligned}$$

The result of the comparison composition is ID only when both actions are equal and DIFF when the actions are different.

Two policies $p_1 = (R_1, \mathfrak{R}_1, E_1, a_{d1})$ and $p_2 = (R_2, \mathfrak{R}_2, E_2, a_{d2})$ are composed by applying the comparison composition $\sim$. The comparison policy $p_1 \sim p_2$ is defined as $(R_1 \cup R_2, \mathbb{R}_{\sim,\mathfrak{R}_1,\mathfrak{R}_2}, \emptyset, a_{d1} \sim a_{d2})$, where the default action is the comparison composition of the two default actions and the comparison resolution strategy $\mathbb{R}_{\sim,\mathfrak{R}_1,\mathfrak{R}_2}$ is defined as:

$$\begin{array}{rccc} \mathbb{R}_{\sim,\mathfrak{R}_1,\mathfrak{R}_2}: & 2^{R_1 \cup R_2} & \longrightarrow & \mathcal{A} \\ & S_1 \cup S_2 & \longmapsto & \mathfrak{R}_1(S_1) \sim \mathfrak{R}_2(S_2) \end{array}$$

The final step is to apply the Semantic-Preserving Morphism to the resulting canonical form. It removes all unnecessary rules form the rule set D and returns a compact representation of the Difference Policy d

10.3 Policy definition

The policy definition is used for the Implementation-verification and defines the desired security behaviour. It is used for both, single policy and multiple policies Implementation-verification In the case of single policy Implementation-verification the policy definition defines the desired security behaviour of the single policy. In the case of multiple policies Implementation-verification the policy definition defines the desired security behaviour of the entire computer network.

The policy definition is expressed in a high-level security policy language. The used high-level language has many similarities with the Geometric-Model and can, therefore, be easily mapped to it. It is defined as an XML document (Listing 10.1). In the high-level language, a policy is composed of a name (`PolicyName`), a type (`PolicyType`), a resolution strategy (`Resolver`), the external data type (`ExternalDataClass`), a default action (`DefAction`) and a set of rules (`Rule`). A rule is composed by an optional external data(`ExternalDataValue`), a set of selectors(`Selector`), an optional transformation(`Transformation`), and an action data(`ActionData`). A rule also has two attributes, the rule action(`Action`) and the rule name(`Label`). The rule action can be set to one of the following values: `ALLOW`, `DENY`, `AH`, `INVERT_AH`, `ESP`, `INVERT_ESP`, `PRENAT`, `POSTNAT`, `LOGGING`. A selector has one attribute (`Label`) that specifies its name, and the only element is the selectors value. The transformation is composed of a list of selectors that define which part of he packet is modified. The action data defines additional information about the action used by the rule.

Listing 10.1: High-level security policy language

```
<!ELEMENT Policy (PolicyName,
                  PolicyType,
                  Resolver,
                  ExternalDataClass,
                  DefAction,
                  Rule*)>

<!ELEMENT PolicyName (#PCDATA)>
<!ELEMENT PolicyType (#PCDATA)>
<!ELEMENT Resolver (#PCDATA)>
<!ELEMENT ExternalDataClass (#PCDATA)>
<!ELEMENT DefAction (#PCDATA)>

<!ELEMENT Rule (ExternalDataValue?,
                Selector+,
                Transformation?,
                ActionData*)>

<!ATTLIST Rule
  Action ( ALLOW | DENY |
           AH | INVERT_AH |
           ESP | INVERT_ESP |
           PRENAT | POSTNAT |
           LOGGING ) #REQUIRED
  Label CDATA #IMPLIED>
<!ELEMENT ExternalDataValue (#PCDATA) >

<!ELEMENT Transformation (Selector+) >

<!ELEMENT ActionData EMPTY>
<!ATTLIST ActionData
  Name CDATA #REQUIRED
  Value CDATA #REQUIRED>

<!ELEMENT Selector (#PCDATA) >
<!ATTLIST Selector
  Label CDATA #REQUIRED>
```

10.4 Application

The comparison model and algorithm presented so far are a general description that does not go into the details of how to apply it to real world scenarios. This section explains how to apply the general comparison model and algorithm to each of the four comparison scopes.

10.4.1 Single Policy Change-impact-analysis

A security policy is modified by the network administrator and the performed changes must be verified. The Single Policy Change-impact-analysis is very similar to the general model and algorithm. Hence, the comparison model is instantiated by setting policy p_a to the original one and policy p_b to the modified one.

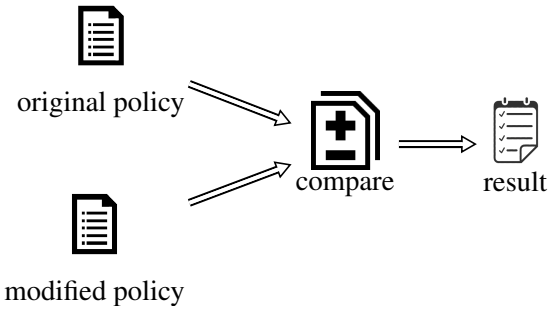


Figure 10.3: Single Policy Change-impact-analysis

$$p_a = \texttt{original policy}$$

$$p_b = \texttt{modified policy}$$

The policy modification can be applied directly to a policy represented in low-level language or represented by the Geometric-Model. Low-Level policies are mapped to the Geometric-Model, which is then given as input to the comparison algorithm.

This comparison scope does not require a precomputed Equivalent-Policy, after all, it also gets no performance advantage from using it. However, it can use the Equivalent-Policy by filtering it the same way as in the Intra-Policy anomaly analysis does.

10.4.2 Single Policy Implementation-verification

The network administrator wants to verify that a policy is correct with respect to the policy definition. The Single Policy Implementation-verification analysis is quite similar to the previous comparison scope. Hence, the comparison model is instantiated by setting policy p_a to the policy in question and policy p_b to the policy definition.

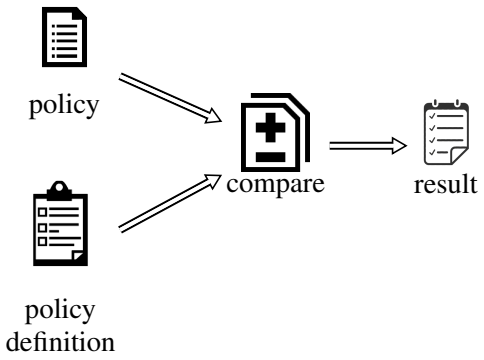


Figure 10.4: Single Policy Implementation-verification

$$p_a = \texttt{policy}$$

$$p_b = \texttt{policy definition}$$

The policy definition can be derived from different sources, depending on the use-case. It can be expressed in different policy abstractions: low-level, Geometric-Model and/or high-level. The high-level policy is the most general use-case, for a security control the desired behaviour is defined and the enforced policy must be verified. The low-level policy is used when a security control is substituted by a never model or from a different vendor and the new policy must enforce the same behaviour as the old one. The Geometric-Model can be used as input when two network administrators have defined a policy and want to compare them to each other. In case the policy definition is expressed in a high-level or low-level language it is mapped to the Geometric-Model.

10.4.3 Multiple Policies Change-impact-analysis

A security policy is modified by the network administrator and the performed changes must be verified with respect to its global behaviour. Therefore, the Equivalent-Policy of the network must be taken inconsideration. The Multiple Policies Change-impact-analysis is at first glance very different to the general comparison model, because in this case multiple policies are involved. However by applying the Equivalent-Policy as description in chapter 7, the application of the comparison model becomes intuitive. For each path, the Equivalent-Policy is computed and, therefore, all policies are united into one Equivalent-Policy that is given as input. Hence, the comparison model is instantiated by setting policy p_a to the original Equivalent-Policy and policy p_b to the modified Equivalent-Policy.

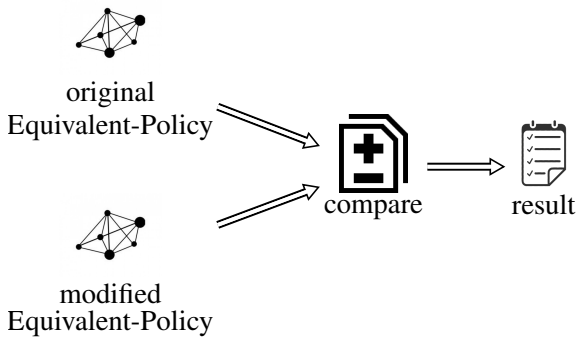


Figure 10.5: Multiple Policy Change-impact-analysis

$$p_a = \texttt{original Equivalent-Policy}$$

$$p_b = \texttt{modified Equivalent-Policy}$$

The policy modification is always performed on the original policies and afterwards, the modified Equivalent-Policy is computed. It is possible to improve the performance of this operation by applying the changes to the Equivalent-Policy instead of completely recomputing it.

10.4.4 Multiple Policies Implementation-verification

The network administrator wants to verify that the global behaviour of a computer network is correct with respect to the policy definition. The Multiple Policies Implementation-verification analysis is a combination of all previous comparison scope. The Equivalent-Policy is calculated for all security policies in question and the policy definition describes the global behaviour of the computer network. Hence, the comparison model is instantiated by setting policy p_a to the Equivalent-Policy in question and policy p_b to the policy definition.

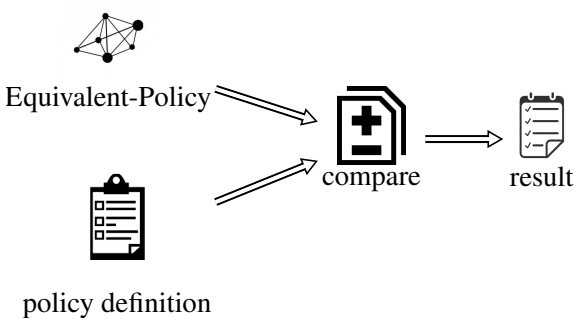


Figure 10.6: Multiple Policy Implementation Verificatio

$$p_a = \texttt{Equivalent-Policy}$$

$$p_b = \texttt{policy definition}$$

Also in this case, the policy definition can be derived from different sources, depending on the use-case. It can be expressed in different policy abstractions: low-level, Geometric-Model and/or high-level.

Part IV

Results

Chapter 11

Implementation

This chapter presents how the Policy Analysis Model has been implemented. The implementation of the Geometric-Model is shown and the generation of the Equivalent-Policy is explained. The implementation of the single analysis modules and their interactions are explained. Furthermore, the API that the implementation exposes is presented. Finally, the Graphical User Interface (GUI) is presented and explained how it supports the administrator by using the Policy Analysis Model.

11.1 Implementation Overview

The Policy Analysis Model has been implemented as a Java library that exposes a well-defined API, and by itself has no user interface. For a representative test and evaluation of the implementation, a GUI has been implemented that uses the API of the Java library.

All essential components of the Policy Analysis Model are modelled by Java classes and interfaces. The most important classes are for security policies, rules, actions, network topologies and the analysis techniques itself. The Java classes expose the required functions for a correct and complete behaviour.

Another important part of the implementation for future development is a comprehensive API. It enables the addition of future extensions to the implementation and also gives the opportunity for integration in already existing tools. The primary functions provided by the API are to read policies from different file formats, to extend the model for new policy types and to execute the different analyses when integrated into an existing tool.

The implemented GUI is based on the API and therefore also proofs that it is complete and usable. A working tool which includes a GUI is also important for empirical tests of the implemented algorithms. The GUI allows the user to define new policies and network

topologies, to execute the different analysis techniques, and to review the results.

Finally, the implementation also includes a complete test-suite to validate the correctness of the algorithms and to perform a performance evaluation. The test-suite includes a network topology generator and security policy generator. The test-suite gives the opportunity to evaluate the solution on a much greater scale without having access to big physical networks. A detailed description of how the security policy generator has been implemented is out of the scope of this work.

11.2 Data Types

This section presents the primary Java classes of the policy analysis library. Figure 11.1 shows the UML class-diagram of the principal classes. The main Java class of the library is the `PolicyAnalysisModel`, it exposes all public API functions. The policy analysis techniques are implemented by the `AnomalyAnalyser`, `ReachabilityAnalser` and `PolicyComparator` class. The `Landscape` class models the network topology and the `Policy` class is used to save the involved security policies. The EquivalentPolicy is computed by the `RuleTransformationResolver` class.

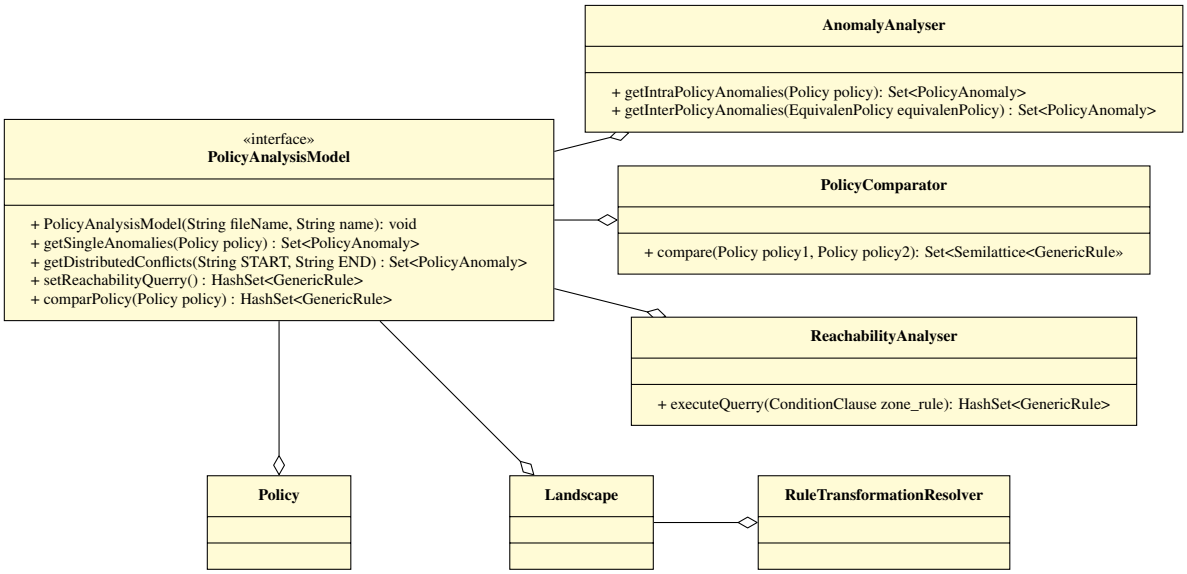


Figure 11.1: UML Data Types

11.2.1 PolicyAnalysisModel

The `PolicyAnalysisModel` class is the main Java class, it exposes all public functions provided by the library. This class is instantiated with a network topology and all associated security policies. It manages the computation of the Equivalent-Policy and caches it for continues usage. The constructor `PolicyAnalysisModel(String fileName, String name)` receives two parameters. The first one is the name of the file that contains all information about the network topology and the security policies. The second one is a symbolic name for the computer network. Furthermore, when a policy analysis technique must be executed, this class calls the appropriate function from the one of the policy analysis classes: `AnomalyAnalyser`, `ReachabilityAnalser` and `PolicyComparator`.

11.2.2 AnomalyAnalyser

The `AnomalyAnalyser` class implements all anomaly analysis techniques and exposes one function for each technique. The function `getIntraPolicyAnomalies()` takes as parameter the security policy to analyse and returns the identified Intra-Policy anomalies. The function `getInterPolicyAnomalies()` takes as parameter the Equivalent-Policy and returns the identified Inter-Policy anomalies.

11.2.3 ReachabilityAnalyser

The `ReachabilityAnalyser` class implements the reachability analysis algorithm and exposes one public function. The constructor `ReachabilityAnalyser()` receives as parameters the network topology and the list of Equivalent-Policies. The function `executeQuery()` takes the reachability query as parameter, and returns the reachability result.

11.2.4 PolicyComparator

The `PolicyComparator` class implements the policy comparison algorithm and exposes one public function. The constructor `PolicyComparator` receives as only parameter the list of selector types. The function `compare()` takes two parameters, the two policies to compare, and returns the comparison result.

11.2.5 Landscape

The `Landscape` class implements all the required functions to represent and manage a network topology. This class contains the list of security controls, their security policies and how they are interconnected. Furthermore, it contains the list of filtering zones and the list of connected hosts. It provides the functions to compute the path from different zones and the traversed security controls.

11.2.6 RuleTransformationResolver

The `RuleTransformationResolver` class exposes one single function that computes the Equivalent-Policy, which is then returned. This function takes as parameters the ordered list of security policies for which the Equivalent-Policy must be computed.

11.2.7 Policy

Figure 11.2 shows the UML class diagram of the interface and classes involving a policy. The `Policy` interface is used by the Java library to model all types of security policies. The UML class diagram includes the most important functions, this interface defines. The function `getPolicyType()` returns the policy type. The function `insertRule()` inserts a new rule into the rule set. There are two versions of this function, one for resolution strategies that do not require an external data and one for those which do. The function `getDefaultAction()` returns the default action of the policy. The function `getResolutionStrategy()` returns the resolution strategy of the policy. The function `match()` returns all rules from the rule set that intersect a given condition clause. The function `evalAction()` returns the action that is enforced by the policy for a given packet. In the implementation, a network packet is represented as a condition clause.

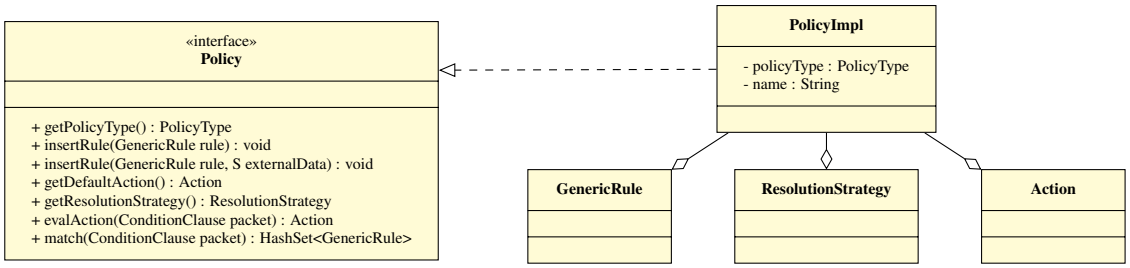


Figure 11.2: UML Policy

The `PolicyImpl` class implements the `Policy` interface and is a basic security policy implementation. The `PolicyImpl` class is composed of a set of `Rules`, a `ResolutionStrategy` and an `Action` as the default action.

11.2.8 ResolutionStrategy

Figure 11.3 shows the UML class diagram regarding the ResolutionStrategy class. The abstract `ResolutionStrategy` class is the base class for all types of resolution strategies. It exposes the most important functions for comparing and composing rules and actions. The function `cloneResolutionStrategy()` returns an exact copy of the resolution strategy. The function `compare()` compares two rules to each other and returns how they are correlated. The function `composeActions()` composes the actions of a set of rules based on the resolution strategy. The function `composeRules()` composes a set of rules and it returns the composed rule. A composed rule is the intersection of the given rules and the enforced action is computed by the `composeActions()` function. The function `isActionEquivalent()` compares the action of two given rules and returns if they are equivalent.

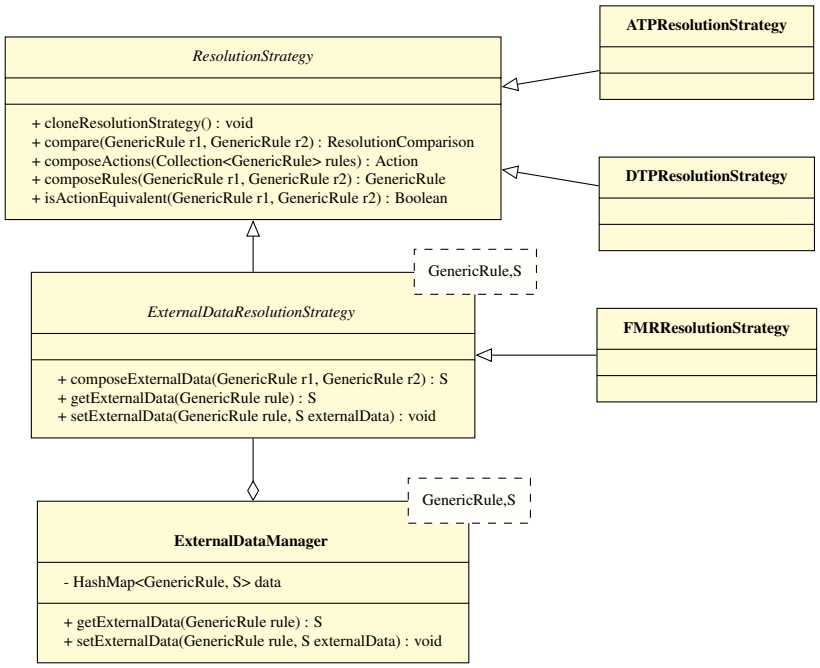


Figure 11.3: UML Resolution Strategy

The two classes `ATPResolutionStrategy` and `DTPResolutionStrategy` extend the abstract class `ResolutionStrategy` and implement real resolution strategies. They enforce the allow-takes-precedence and deny-takes-precedence resolution strategies respectively. The `ResolutionStrategy` class does not consider external data. Therefore, the abstract class `ExternalDataResolutionStrategy` extends it and provides the support for external data. The function `composeExternalData()` composes the external data of two given rules. The function `getExternalData()` returns the external data associated with a given rule. The function `setExternalData()` associates the external data to a rule. The class `ExternalDataManager` stores and manages the associations between rules and external data. The FMR resolution strategy is implemented by the `FMRResolutionStrategy` class.

11.2.9 GenericRule

The `GenericRule` models policy rules and is used by the `Policy` class. This class is composed by the `Action` class and the `ConditionClause`. The `Action` represents the enforced rule action and the `ConditionClause` represents the rule condition.

Figure 11.4 shows the UML class diagram for the `GenericRule` class. It exposes functions that evaluate the correlation between rules. The function `getAction()` returns the action enforced by the rule. The function `getConditionClause()` returns the rule condition. The function `isIntersecting()` evaluates if the rule intersects a given rule. The function `ruleClone()` creates an exact copy of the rule and returns it.

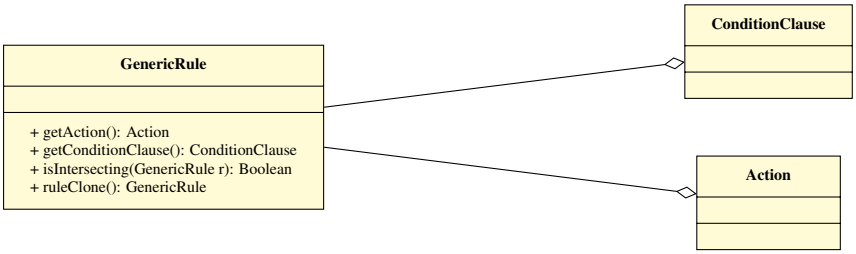


Figure 11.4: UML GenericRule

11.2.10 ConditionClause

The `ConditionClause` class contains a list of `Selector` to model the different dimensions of the hyper-rectangle. It exposes functions that provide the functionality to compose condition clauses and to evaluate the correlation between them. Figure 11.5 shows the UML class diagram of the `ConditionClause` class. The function `conditionClauseClone()` creates a complete copy of the condition clause. The function `intersection()` computes the intersection with a given condition clause. The function `isConditionEquivalent()` evaluates if the given condition clause is equivalent. The function `isConditionSubset()` evaluates if the given condition clause is a subset. The function `isConditionSubsetOrEquivalent()` evaluates if the given condition clause is a subset or equivalent. The function `isIntersecting()` evaluates if the given condition clause is intersecting. The function `addSelector()` inserts the given selector into the selector list. The function `setSelector()` substitutes a given selector in the selector list.

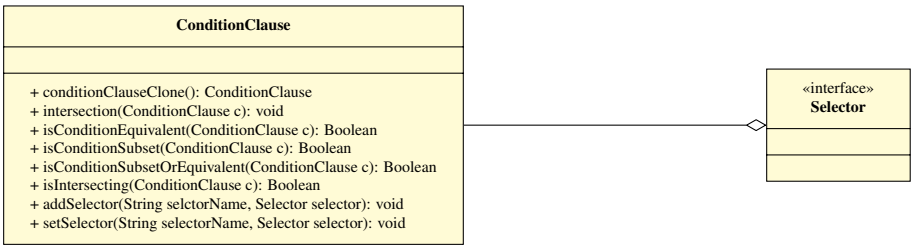


Figure 11.5: UML ConditionClause

11.2.11 Selector

The `ConditionClause` interface is implemented by the different selector implementations. It exposes functions that are used to compute the intersection of two selectors and to verify the correlation between them. The function `intersect()` computes the intersection of the selector with a given selector. The function `isEquivalent()` evaluates if the selector is equivalent to a given selector. The function `isSubset()` evaluates if the selector is a subset of a given selector. The function `isSubsetOrEquivalent()` evaluates if the selector is a subset of or equivalent to a given selector. The function `isIntersecting()` evaluates if the selector intersects the given selector. For each type of packet field, there exists a selector class.

For example, IP addresses are modelled by the `IpSelector` class, a port number is modelled by the `PortSelector` class and a protocol type is modelled by the `ProtocolSelector`.

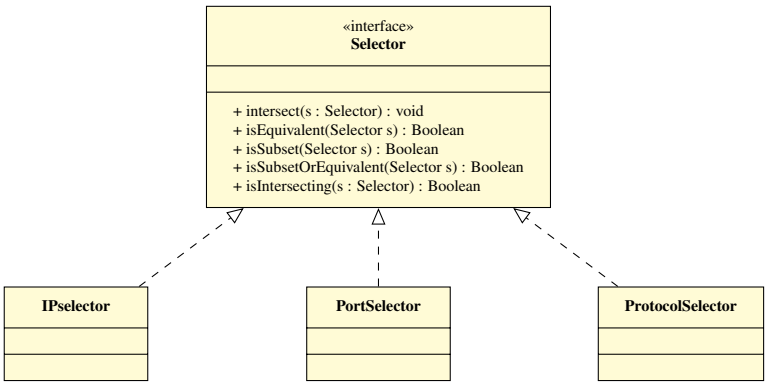


Figure 11.6: UML Selector

11.2.12 Action

The `Action` interface is implemented by the different action classes. It exposes very few functions because a rule action, in general, is a very simple structure. The most important function is the `actionClone()` function, it creates an exact copy of the action. The `FilteringAction` class, for example, is implemented as an enum that contains the filtering actions ALLOW and DENY. The `TransformatonAction` class is an abstract class to model more complex transformation actions such as NAT or IPsec. The transformation condition is accessible by the function `getTransformation()`. This class contains a `ConditionClause` that represents the transformation applied to a packet. The `IPSecAction` class is used to represent IPsec actions and, therefore, contains also encryption and authentication parameters. The function `getKey()` returns the IPsec key. The function `getHashKey()` returns the IPsec key hash. The function `getType()` returns the IPsec type, AH or ESP. The function `isEqual()` evaluates if the IPsec action is equivalent to a given IPsec action. The `NATAction` class is used to represent NAT actions and, therefore, contains also the NAT action type, PRENAT or POSTNAT. The function `getNATAction()` returns the NAT action type. Figure 11.7 shows the UML class diagram for this classes and interfaces.

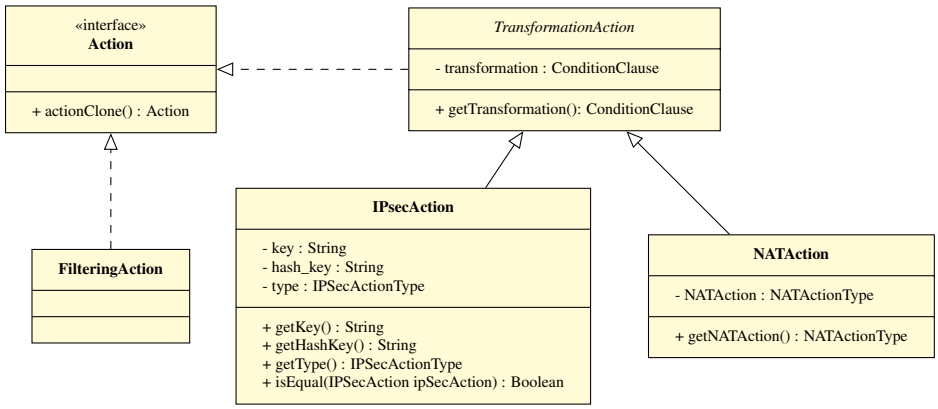


Figure 11.7: UML Action

11.3 Graphical User Interface

The Graphical User Interface (GUI) is based on the Eclipse Rich Client Platform (RCP) and uses the API exposed by the Analysis Model Library. The GUI provides additional features to improve the usability of the tool. It can save and load a computer network topology and the associated security policies to and from a file. It also provides a policy and network topology editor. All analysis techniques can be executed and the results are presented within the GUI.

11.3.1 Editor

Figure 11.8 shows the network topology editor. It can be used to inspect the topology and allows to add new security controls, hosts and filtering zones.

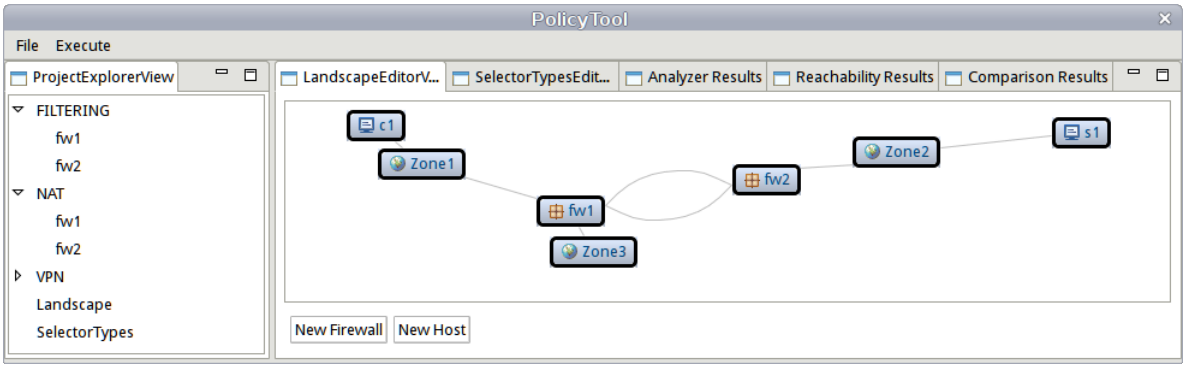


Figure 11.8: Network Topology Editor.

Figure 11.9 shows the policy editor. It can be used to inspect the security policy and allows to add, modify and delete policy rules. When a new rule gets inserted, the *New Rule* wizard is shown. When a rule gets modified the *Modify Rule* wizard is shown.

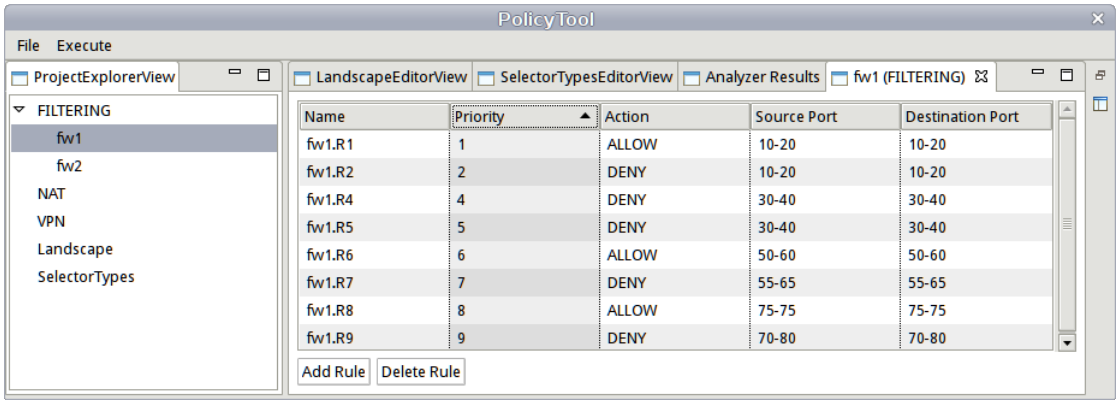
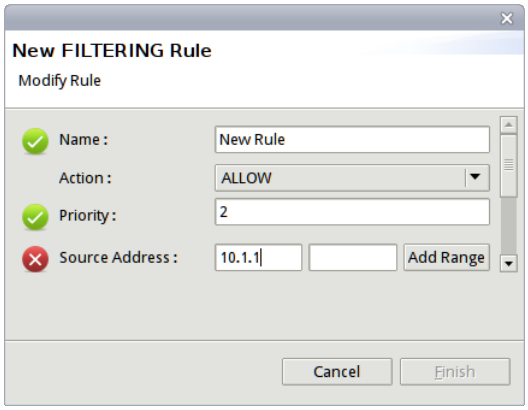
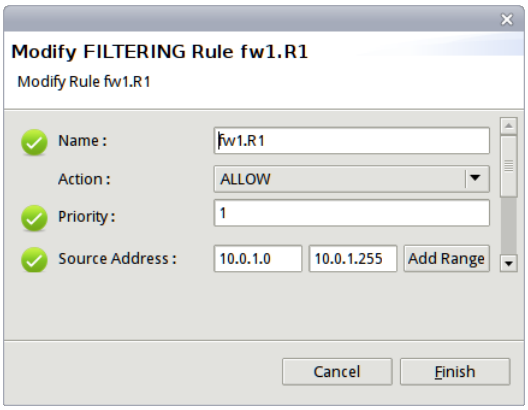


Figure 11.9: Policy Editor Window.

Figure 11.10 shows the New Rule Wizard and the Modify Rule Wizard. The New Rule Wizard enables the user to define a new rule based on all possible selectors. The Modify Rule Wizard shows all defined selectors of the rule that can be modified or deleted. Furthermore, new selectors can be added to the rule. Both wizards verify the input of the user and only valid values are accepted. Non-valid values are highlighted and the user is alerted.



(a) New Rule Wizard.



(b) Modify Rule Wizard.

Figure 11.10: Rule editor.

11.3.2 Analysis Execution

The different policy analysis techniques can be performed by selecting the appropriated menu point. Figure 11.11 shows the different possibilities.

The Intra-Policy anomaly analysis is performed by right-clicking on the policy name in the *PolicyExplorerWindow* and selecting the menu point *Analyse* from the drop-down menu. The result is then presented in the *Analyser Result* tap.

The Inter-Policy anomaly analysis is performed by selecting the menu point *Distributed Analysis* from the *Execute* menu. In the appearing wizard, the user must select the source and destination zone for the analysis. The result is then presented in the *Analyser Result* tap.

The reachability analysis is performed by selecting the menu point *Reachability Analysis* from the *Execute* menu. In the appearing wizard, the user must specify the reachability query he wants to execute. The result is then presented in the *Reachability Result* tap.

The Policy Comparison is performed by selecting the menu point *Policy Comparison* from the *Execute* menu. In the appearing wizard, the user must select the policy he wants to compare. The result is then presented in the *Comparison Result* tap.

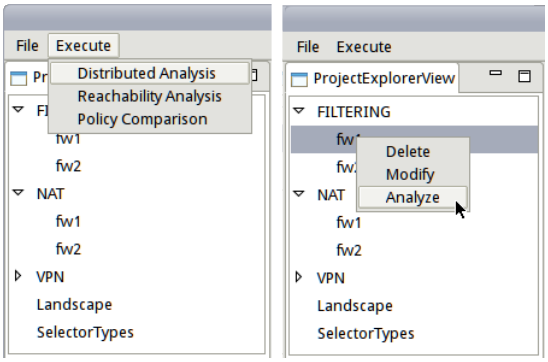


Figure 11.11: Policy Analysis Execution.

11.3.3 Result representation

Figure 11.12 shows the result window of the Intra-Policy analysis. When the mouse hovers over an anomaly the involved rule with all its selectors and enforced action is shown. The result window for the Inter-Policy analysis has the same structure. Figure 11.13 shows the result window of a reachability analysis. Figure 11.14 shows the result window of a policy comparison.

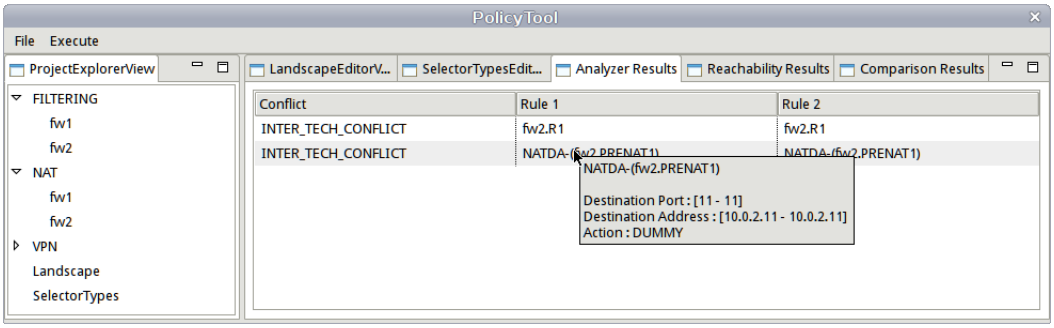


Figure 11.12: Intra-Policy Analysis Result Window.

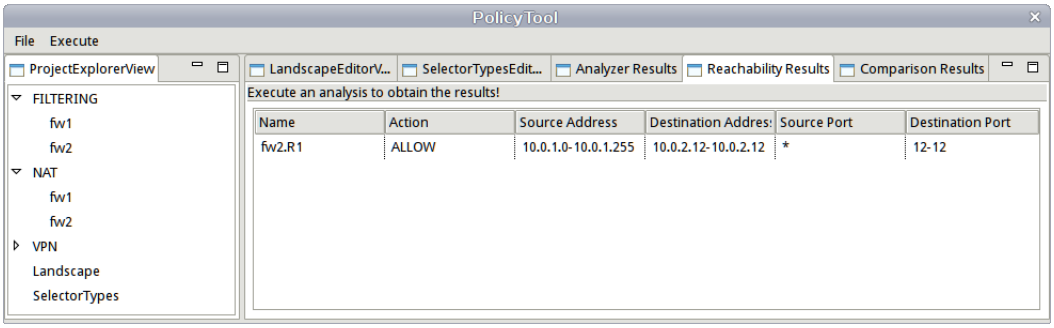


Figure 11.13: Reachability Analysis Result Window.

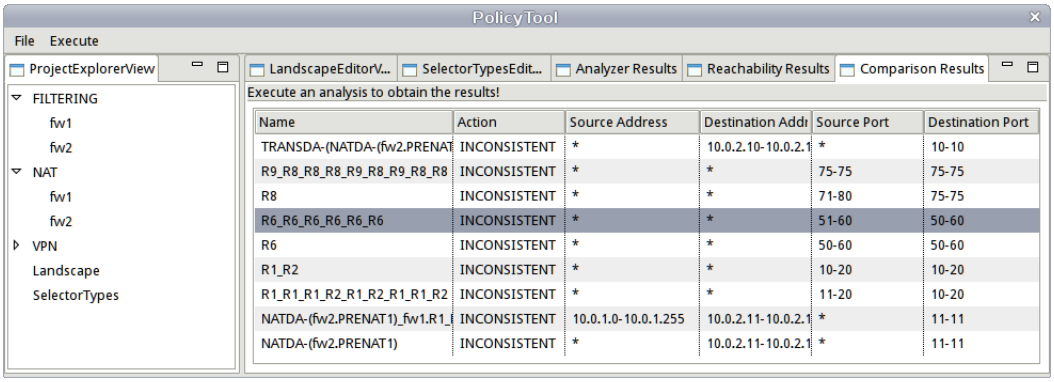


Figure 11.14: Policy Comparison Result Window.

Chapter 12

Validation

This chapter presents the performance evaluation of the Policy Analysis Model implementation. First the test environment is presented, in particular, the test hardware and the test cases. There have been performed two test cases, one on a synthetic network and one on a campus network. Each of the test cases is described, and the performance results are presented.

12.1 Test environment

The performance tests evaluate the execution time of the different policy analysis techniques. The tests are designed to cover the greatest number of possible network topologies and configurations. The first performance test is performed on several synthetic networks to prove scalability of the model. The second performance test is performed on the representation of a campus network to prove real world application. The synthetic networks are generated for different numbers of serially connected security controls. A security policy generator was used to generate the policies for the synthetic networks. The security policy generator generates security policies according to real statistical properties. Furthermore, it also takes into consideration the network topology, for which the policy is generated, to enable end-to-end reachability. The test on the campus network proofs the validity of the analysis model for real networks. The campus network has a reasonable size to be representative. Furthermore, it also contains different security policy categories.

As already presented in the previous chapter, the analysis model has been implemented in Java 1.6. All test have been executed on OpenJDK Runtime Environment (IcedTea 2.5.3). The test environment is based on an Intel Core i7-3630QM (2.4 GHz) CPU, with 16 GB RAM under Debian Linux 8 operating system.

12.2 Synthetic network

The synthetic network is composed of a variable number of serially connected security controls. Each security control is configured by a security policy containing a variable number of policy rules. The different types of security policies are applied: filtering, communication protection, logging, transformation and routing. In particular, three independent variable parameters characterize the synthetic network:

- sc, the number of security controls on each path, variable from 1 to 5;
- r, the average number of rules for each security policy, variable from 50 to 500;
- t, the number of transformation and communication protection rules for each path, which ranges from 1 to 25.

The most complex synthetic network is composed of five security controls with 500 rules each and 25 transformation and communication protection rules. According to a survey by Algosec [98] security policies include in general less than 200 rules. The synthetic network should represent very well a real worst case and, therefore, is the performance evaluation representative.

The performance tests have been executed 100 times for each synthetic network, and the presented results are the average of them. Additionally, the 95% confidence interval has been computed and its also shown in the graphs as a grey area around the graph curves.

12.2.1 Security Policy Generation

Implementing the security policy generation is no easy task, different aspects must be taken into consideration. One very important aspect is the end-to-end communication in case the generated policies are interconnected. Therefore, the security policy generation takes this into consideration and specific "allow rules" are inserted into the policies. Furthermore, all policies are generated according to the statistical properties presented by Taylor [99] with an average number of anomalies according to Al-Shaer [100]. The survey work reported that that no more than 25% of rules intersect each other and that at most five rules intersect simultaneously. Therefore, security policies with such statistical properties should represent very well actual policies and, therefore, allow a representative performance evaluation.

12.2.2 Equivalent-Policy creation

The Equivalent-Policy creation is the first task to perform because the all policy analysis techniques are performed on it. This subsection presents the performance evaluation of the Equivalent-Policy creation depending on different factors. First, the time to create the Equivalent-Policy is evaluated. Second, the size of the Equivalent-Policy is evaluated. And finally, the dependency between the final size and the creation time.

Equivalent-Policy creation time

First, it will be evaluated how long it takes to compute the Equivalent-Policy depending on the number of security controls, the average number of rules and the number of transformation and communication protection rules. The time to compute the Equivalent-Policy of a synthetic network based on different parameters is shown in Figure 12.1 and Figure 12.2.

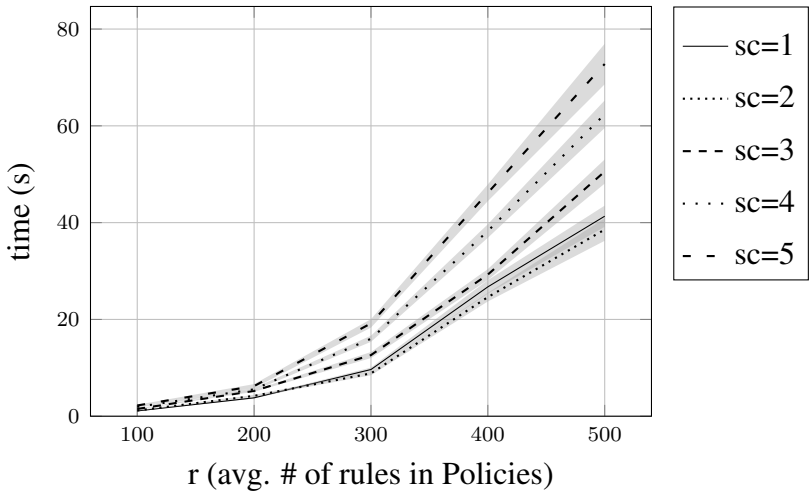


Figure 12.1: Time to compute a Equivalent-Policy depending on sc and r with $t = 25$.

Figure 12.1 shows the time required to compute the Equivalent-Policy for different numbers of rules and security controls. The average number of rules in the security policies varies from 100 to 500 and is reported on the x-axis of the graph. The number of security controls varies from 1 to 5 and is reported as different plots within the graph. The number of transformation and communication protection rules is fixed at 25 for all tests.

Figure 12.2 shows the time required to compute the Equivalent-Policy for different numbers of rules and transformation and communication protection rules. The average number of rules in the security policies varies from 100 to 500 and is reported on the x-axis of the graph. The average number of transformation and communication protection rules in the security policies varies from 5 to 25 and is reported as different plots within the graph. The number of security controls is fixed at 5 for all tests.

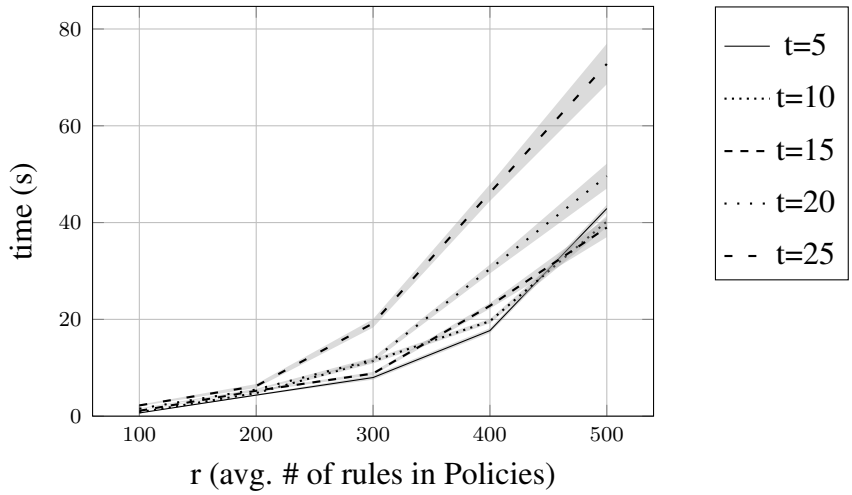


Figure 12.2: Time to compute a Equivalent-Policy depending on t and r with $sc = 5$.

The number of security controls on the path (Figure 12.1) and the number of transformation and communication protection rules per path (Figure 12.2) slightly more than a linear influence on the computation time. The number of rules has the biggest impact on the computation time. The Equivalent-Policy computation takes about 70 s in the worst case. Considering that the Equivalent-Policy must be computed only once, these results are very promising.

Furthermore, by also taking into consideration that the average number of rules is bigger than the real security policies, this result are more than sufficient. In general a security control has less than 200 rules [98] and in this case, the computation takes less than 10 s. Furthermore, the number of security controls and the number of transformation and communication protection rules have very little impact when working with less than 200 rules per security policy. Therefore, the performance of the Equivalent-Policy creation is very good.

Equivalent-Policy size

The Equivalent-Policy is generated starting from interconnected security policies by substituting all transformation and channel protection policies. Furthermore, redundant and unnecessary rules are removed from the Equivalent-Policy. The following tests evaluate the number of rules in the Equivalent-Policy depending on the number of security controls, the number of rules and the number of transformation and communication protection rules. The number of rules in the Equivalent-Policy of a synthetic network based on different parameters is shown in Figure 12.3 and Figure 12.4

Figure 12.3 shows the number of rules in the Equivalent-Policy for different numbers of rules and security controls. The average number of rules in the security policies varies from 100 to 500 and is reported on the x-axis of the graph. The number of security controls varies from 1 to 5 and is reported as different plots within the graph. The number of transformation and communication protection rules is fixed at 25 for all tests.

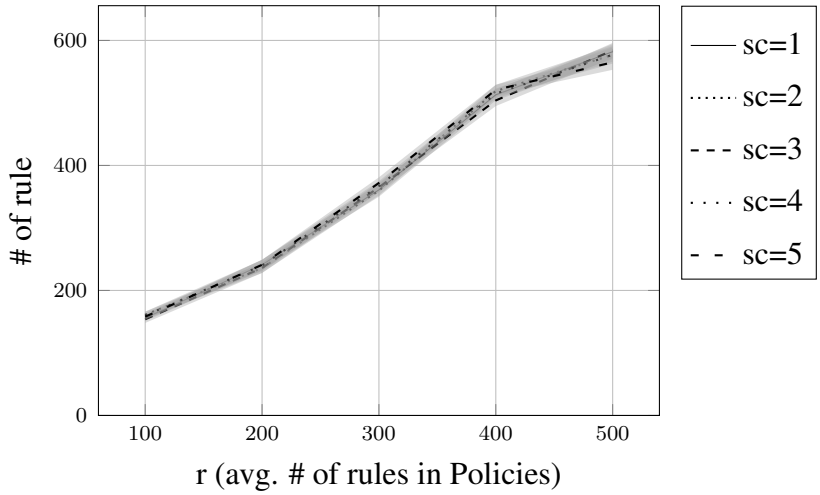


Figure 12.3: Number of rules in Equivalent-Policy depending on sc and r with $t = 25$.

The number of security controls (Figure 12.3) has essentially no impact on the number of rules in the Equivalent-Policy. They are relatively independent of each other because all rules in the security controls are very similar to allow end-to-end communication. Therefore, the policy composition includes many redundant rules which are removed from the Equivalent-Policy.

Figure 12.4 shows the number of rules in the Equivalent-Policy for different numbers of rules and transformation and communication protection rules. The average number of rules in the security policies varies from 100 to 500 and is reported on the x-axis of the graph. The average number of transformation and communication protection rules in the security policies varies from 5 to 25 and is reported as different plots within the graph. The number of security controls is fixed at 5 for all tests.

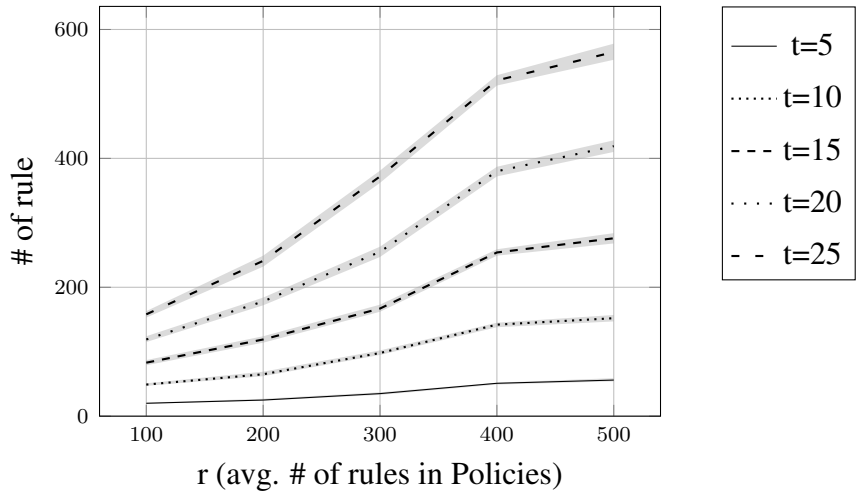


Figure 12.4: Number of rules in Equivalent-Policy depending on t and r with $sc = 5$.

The number of transformation and communication protection rules (Figure 12.4) has a greater impact on the number of rules in the Equivalent-Policy. Each transformation and communication protection rules is transformed during the Equivalent-Policy creation, and new policy rules are inserted. Therefore, the overall size of the Equivalent-Policy increases. However, it can be expected that the number of transformation and communication protection rules is limited.

In summary, the overall size of the Equivalent-Policy is manly influenced by the average number of rules and the number of transformation and communication protection rules. However, this dependency does not affect the scalability of the model significantly because the number of security controls does not affect at all the size of the Equivalent-Policy.

Equivalent-Policy size-time

The last test for evaluating the Equivalent-Policy computation compares the creation time with the number of rules included in the Equivalent-Policy. The synthetic networks, used for the tests, contain a variable number of security controls from 1 to 5, a variable number of rules from 50 to 500 and a variable number of transformation and communication protection rules for each path from 1 to 25.

Figure 12.5 shows the time required to compute a Equivalent-Policy of a certain size. The x-axis represents the number of rules in the Equivalent-Policy and the y-axis represents the computation time. The computation time has a nearly linear dependency on the number of rules in the Equivalent-Policy. This test has a greater confidence interval because the computation time depends not only on the number of rules in the Equivalent-Policy.

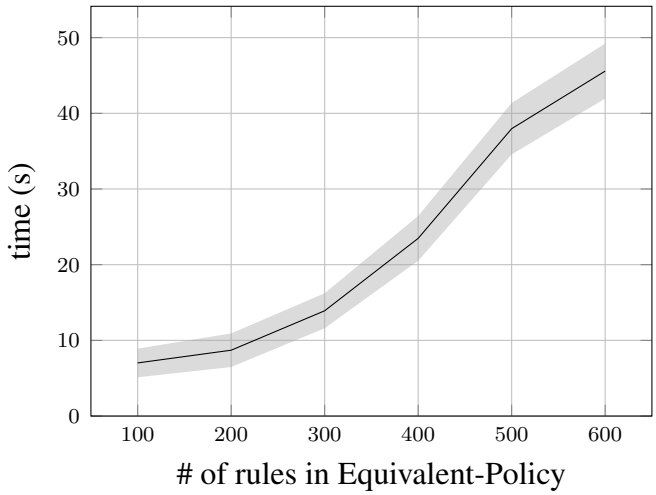


Figure 12.5: Time to compute Equivalent-Policy based on number of rules.

These test results are mainly necessary for the evaluation of the performance test for the analysis techniques. The different analysis techniques are preformed on precomputed Equivalent-Policies. Therefore, they are directly correlated with this tests.

12.2.3 Anomaly Analysis

The anomaly analysis is performed on a precomputed Equivalent-Policy. Therefore, the performance evaluation of the anomaly analysis is based on the number of rules in the Equivalent-Policy.

Figure 12.6 plots the average time required to perform the anomaly analysis based on the different number of rules in the Equivalent-Policy. The number of identified anomalies is not reported because they are not statistical representative.

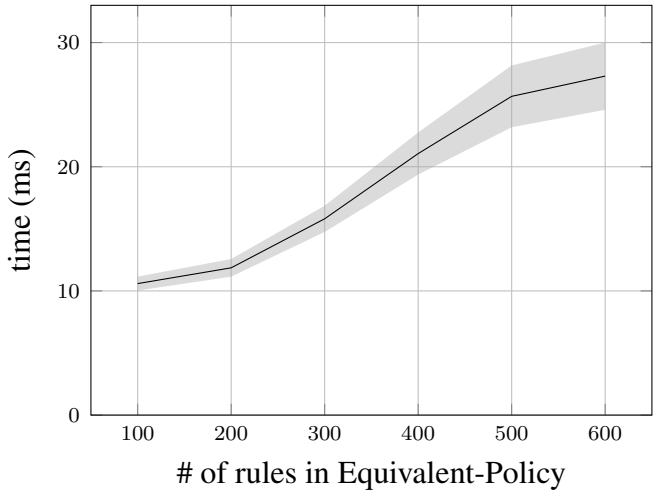


Figure 12.6: Time to perform an Anomaly Analysis.

The time required to perform the analysis has a linear dependency on the number of rules in the Equivalent-Policy. In the worst case tested the execution time was less than 30 milliseconds. The confidence interval increases with the number of rules because the standard deviation of the number of identified anomalies is correlated with the number of rules. The correlation between the number of rules and the standard deviation of the number of identified anomalies exists only because the rules are randomly generated. However, even the upper bound of the confidence interval has a sufficient performance. Therefore, the imprecisions due to the random generation can be ignored.

12.2.4 Reachability Analysis

The reachability analysis is performed on a precomputed Equivalent-Policy. Therefore, the performance evaluation of the reachability analysis is based on the number of rules in the Equivalent-Policy.

Figure 12.7 plots the average time required to perform a reachability analysis based on the different number of rules in the Equivalent-Policy. Three distinct reachability queries have been performed: host-host, host-zone and zone-zone. The first query (1-1) evaluates the reachability between two hosts. The second query (1-x) evaluates the reachability between a host and the rest of the network. The third query (x-x) evaluates the reachability of the entire network.

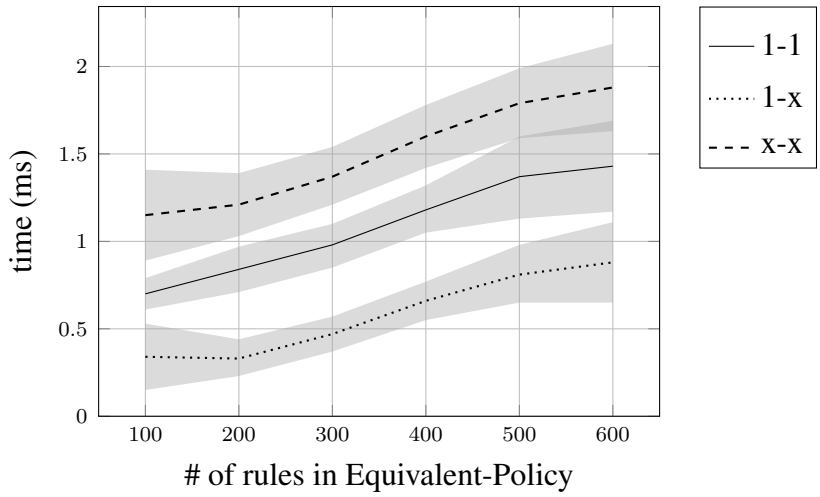


Figure 12.7: Time to execute a reachability query.

The time required to execute the query has a linear dependency on the number of rules in the Equivalent-Policy. In the worst case tested the execution time was less than two milliseconds. The confidence interval is relatively big because of the measurement imprecisions. The measurements have a precision of one millisecond, and because the results are also in this range, they are not very precise. However, the measurement imprecisions are not important in this case because the test results are very good even from a worst-case perspective.

12.2.5 Policy Comparison

The policy comparison is performed on a precomputed Equivalent-Policy. Therefore, the performance evaluation of the policy comparison is based on the number of rules in the Equivalent-Policy.

Figure 12.8 plots the average time to perform a policy comparison based on the different number of rules in the Equivalent-Policy. Three comparisons with various numbers of differences have been performed: The first comparison (# 0) compares the network configuration with itself and, therefore, identifies no differences. The second comparison (# 1) compares the network configuration with one modification. The third comparison (# 10) compares the network configuration with ten modification.

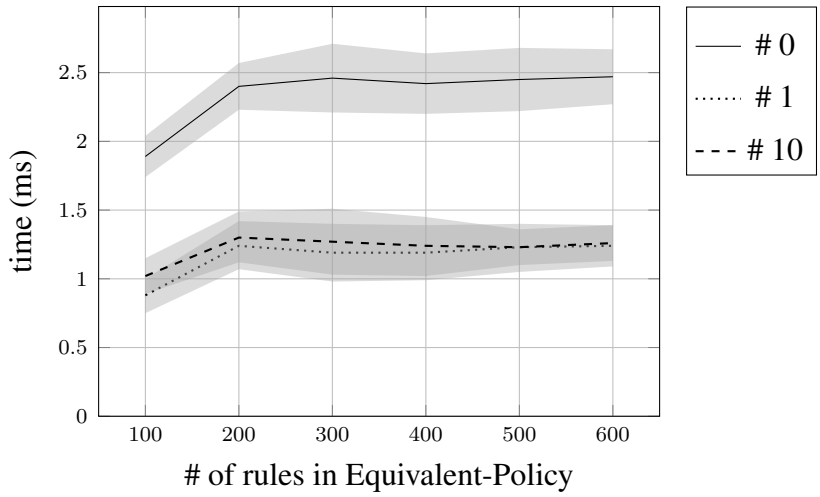


Figure 12.8: Time to perform a policy comparison.

The time required to perform the comparison is nearly constant and is independent of the number of rules in the Equivalent-Policy. In the worst case tested the execution time was less than three milliseconds. Also, in this case, the confidence interval is relatively big because of the measurement imprecisions. The measurements have a precision of one millisecond, and because the results are also in this range, they are not very precise. However, the measurement imprecisions are not important in this case because the test results are very good even from a worst-case perspective.

12.3 Campus network

The second set of test has been performed on a representation of a campus network. These tests validate the performance of the proposed model on a real network. Furthermore, the results of the tests also prove that the synthetic network tests are representative.

The campus network is composed of 19 security controls, one router, three NAT devices, 12 firewalls with NAT functionality, one firewall with IPSec VPN functionality and two packet-filter firewalls. There are about 10,000 connected hosts, subdivided into 36 filtering zones with private IP addresses and 30 additional zones with public IP addresses. All filtering zones with private IP addresses are connected through a NAT devices with the rest of the network, where each NAT device handles 12 filtering zones.

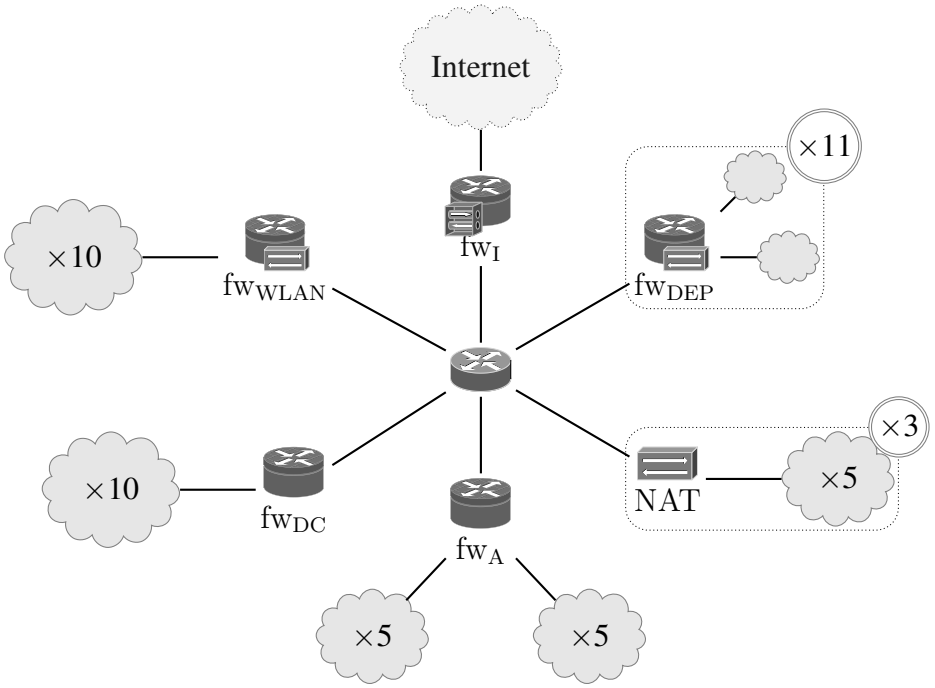


Figure 12.9: The campus network.

Figure 12.9 shows the network topology of the campus network and all its connected security controls and filtering zones. The network topology is a star topology with one central core router. Therefore, each filtering zone is interconnected through three security devices to each other filtering zone in the network.

In detail the campus network is composed of the following security controls:

- The core router is the center of the star topology, it is connected to the Internet through the border firewall $\mathrm{fw_I}$.
- The border firewall $\mathrm{fw_I}$ is configured by a security policy with about 150 rules, it also provides VPN functionalities for external remote connections.
- The campus network includes 11 department networks, where each department networks consists of two filtering zones: a private and a public zone. The private zone is connected to the core router by means of a NAT/NAPT and a firewall ($\mathrm{fw_{DIP}}$) and can only connect to the rest of the campus network. The public zone is connected to the core router by just the firewall ($\mathrm{fw_{DIP}}$) and is reachable from the Internet.
- The campus network includes three laboratory networks composed of five filtering zones each. Laboratory network is connected to the core router via an individual NAT/NAPT device (NAT). These networks are allowed to access the campus network, and the Internet, but not incoming connections from the Internet are permitted.
- The campus network includes one administration network that is composed of five filtering zones for servers and five filtering zones for the employees' workstations. All filtering zones are connected to the core router by the firewall ($\mathrm{fw_A}$) that contains approximately 250 rules. The five filtering zones for servers have higher security requirements than the other ones and are accessed mainly by the users from the five administration user zones.
- The campus network includes one data center that is composed of ten filtering zones. These filtering zones contain various servers (web, e-mail, file servers, ...), some of which are also accessible from the Internet. The data-center is connected to the core router by a single firewall ($\mathrm{fw_{DC}}$) with about 50 rules.
- The campus network includes ten wireless networks that are connected to the core router by one firewall ($\mathrm{fw_{WLAN}}$) with about 20 rules (acting as NAT/NAPT and firewall). Wireless clients can access the Internet. However, they can only access the public parts of the campus network.

The performance evaluation on the campus network has been executed in two steps. First, the Equivalent-Policy for all 4624 possible paths has been computed. Afterwards, the different analysis techniques have been applied. The anomaly analysis applied searches for Inter-policy anomalies within the entire network. The reachability analysis that has been applied computes the reachability parameters of the entire network. The policy comparison applied compares the network with itself and without modifications.

The computation of the Equivalent-Policy took about 487 seconds. Considering the network size it is very efficient, and compared to the state of the art has a better performance. The anomaly analysis took about 14 milliseconds and can therefore be used for real-time execution. The reachability analysis took about 83 milliseconds to compute the reachability parameters of the entire network. The policy comparison took about 696 milliseconds, and is therefore the computational most intensive task. These results proof that the performance of the proposed model is adequate to be applied to real networks.

Although, the actual security policies of the campus network were not used it is still a valid performance evaluation because the network topology, the size and policy statistics are the equivalent. The security policies have been generated by the security policy generator according to the real policy statistics. Furthermore, the filtering zones in the network topology are interconnected through different paths and therefore through different policy transformations.

Part V

Conclusion

Chapter 13

Conclusion

This work has proposed an innovative security policy analysis model that combines different policy analysis techniques. The supported policy analysis techniques are: anomaly analysis, reachability analysis and policy comparison. This work has proposed the first model in the state of the art that combines all three analysis techniques. It is the only one that covers the complete analysis spectrum and supports the most use-cases.

Anomaly analysis searches for potential misconfigurations within one policy (Intra-Policy analysis) or between multiple policies (Inter-Policy analysis). Policy analysis has been studied for over a decade and is a very promising research area. This work improves the state of the art even further by introducing a new anomaly classification and the support for new security controls. The proposed anomaly classification also includes a new anomaly class, the Transformed Traffic anomalies. A Transformed Traffic anomaly occurs when a packet can not be evaluated by a security policy because it was modified or encrypted by a previous security policy. Furthermore, the analysis algorithm is improved by using the Geometric-Model, which allows a more detailed analysis. The new model can identify rule anomalies not only among rule pairs but also among an entire rule set.

Reachability analysis verifies specific properties within a security policy or an entire computer network. It can be performed online by injecting test packets into a deployed network or offline by querying a model. This work presents an offline reachability analysis that supports the most important security policy types. The main improvement over the state of the art is the improved expressiveness of the query language and the result representation. The query language supports different reachability and connection types. The result representation is also an important aspect of the performed research. The reachability analysis result is structured like a security policy and therefore well readable by network administrators. Furthermore, the proposed model is extensible to future security policy types.

Policy comparison is used to compare one or more security policies to each other. There are two main use-cases: Implementation-verification and Change-impact-analysis. Implementation-verification evaluates that a high-level policy is correctly implemented. Change-impact-analysis evaluates a modification that has been performed on a security policy or the network topology. The proposed model supports both types, Implementation-verification and Change-impact-analysis. Furthermore, the model is not only able to compare single policies but also entire computer network configurations. The solutions in the state of the art support only one use-case, whereas the proposed model supports all four use-cases. The proposed model can therefore be applied to the most use-cases.

A central feature of the proposed analysis model is an innovative concept called Equivalent-Policy. The Equivalent-Policy is a compact representation of the entire computer network with all its security policies. The Equivalent-Policy is computed by following three steps: Firstly, substituting all transformation and channel protection policies. Secondly, composing the remaining security policies. Thirdly, eliminating all redundant and unnecessary rules. It supports the following security policy types: packet-filter, stateful, Layer7, Routing, IPsec, SSL, NAT/NAPT, Web-Proxies and Monitoring. Furthermore, the Equivalent-Policy is designed to be extensible for future security policy types. The Equivalent-Policy is the key component because all policy analysis techniques are performed on it.

The proposed model has several advantages and improvements over the state of the art. First of all, it is the first one to include all three analysis techniques. Secondly, each analysis technique is improves the state of the art. Thirdly, the model covers the most security policy types. Fourthly, it is extensible and easy to adapt to new types of security policies.

The proposed model was implemented as a Java prototype to prove its validity. Furthermore, a GUI was implemented for ease of use and to show that the prototype could be extended into legitimate software. Extensive tests was performed on the implementation to validate its performance. The results show that the proposed model is valid and could be used to analyse real computer networks. Therefore, this model could be integrated into future network architectures such as VNF/SDN. The implemented prototype is a good starting point for such improvements because it is a library with a well-defined API.

Although the proposed model is a major improvement over the state of the art and also its performance evaluation was successful, it can still be improved in future work. First of all, the functionalities of the model can be extended by incorporating an anomaly resolution algorithm. Secondly, the expressiveness of the high-level policy language can be improved. And last but not least, new security policy types can be integrated and potentially new anomaly categories identified.

An anomaly resolution algorithm has the advantage that no human interaction during the resolution of the identified anomalies is required. In the state of the art there have already been some attempts to for such an algorithm. These algorithms are not very advanced and focus primarily on filtering policies. However, they can be the foundation for future research. The first step could be to integrate these algorithms into the proposed model. Afterwards, they can be improved and extended to support other security policies.

The high-level security policy language is mainly important for policy comparison when performing Implementation-verification. In the current state the high-level language, used for the policy definition, can express the most important security aspects of a computer network. However, it is still very technical and requires a decent background knowledge in computer network security. Future research could focus on defining an improved version that is closer to human language and, therefore, easier to use by network administrators. Also non tech-savvy people could use the tool when a policy language is provided that is closer to human language.

The proposed model is designed for future extensions and integration of new security policy types. Therefore, this advantage should be exploited, and the integration of additional security policy types should be studied.

Part VI

Appendix

Abbreviations and Symbols

Abbreviations

ACL	Access Control List
AH	Authentication Header
ATP	Allow Takes Precedence
BDD	Binary Decision Diagram
CAN	Canonical Form
D.Port	Destination Port
DC	Data-Center
DEP	Department
DIFF	Different
DTP	Deny Takes Precedence
eqPolicy	Equivalent-Policy
ESP	Encapsulating Security Payload
FAME	Firewall Anomaly Management Environment
FANG	Firewall Analysis Engine
FDD	Firewall Decision Diagram
FMR	First Matching Rule
FTP	File Transfer Protocol

FW	Firewall
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
ID	Identical
IDD	Interval Decision Diagram
IP	Internet Protocol
IPsec	Internet Protocol Security
ISO	International Organization for Standardization
IT	Information Technology
LUB	Least Upper Bound
MDD	Multi-way Decision Diagram
MSTP	Most Specific Takes Precedence
NAPT	Network Address and Port Translation
NAT	Network Address Translation
NFV	Network Functions Virtualization
NIDS	Network Intrusion Detection System
OBDD	Ordered Binary Decision Diagrams
OSI	Open Systems Interconnection
PCI-DSS	Payment Card Industry Data Security Standard
RCP	Rich Client Platform
S.Port	Source Port
SAT	Boolean Satisfiability Problem
SC	Security Control

SDN	Software-Defined Networks
SFQL	Structured Firewall Query Language
SRQL	Structured Reachability Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Uniform Resource Identifier
VPN	Virtual Private Network
WSS	WS-Security, Web Services Security
XML	Extensible Markup Language

Symbols

$\mathbb{F}_i$	Packet field
$\mathbb{P} = \{\mathbb{F}_1, \ldots, \mathbb{F}_n\}$	Packet space
$x = \{f_i \in \mathbb{F}_i,\ \mathbb{F}_i \in \mathbb{P}_I,\ I \subseteq [1, n]\}$	Packet
$\mathcal{C} = \prod_{i \in I} \mathbb{F}_i \qquad \mathbb{F}_i \in \mathbb{P}_I \qquad I \subseteq [1, m]$	Selection space
s_i	Selector
$c = \prod_{i \in I} s_i \qquad s_i \subseteq \mathbb{F}_i \qquad I \subseteq [1, m] c \in \mathcal{C}$	Condition
$a : \mathbb{P} \to \mathbb{P} \cup \emptyset a \in \mathcal{A}$	Action
$\mathcal{A}$	Action space
σ_i	Action transformation value
$\gamma = \prod_{j \in I_\mathrm{T}} \sigma_j \qquad \sigma_j \subseteq \mathbb{F}_j \qquad I_\mathrm{T} \subseteq [1, m]$	Action clause
$\mathcal{T} = \prod_{j \in I_\mathrm{T}} \mathbb{F}_j \qquad \mathbb{F}_j \in \mathbb{P} \qquad I_\mathrm{T} \subseteq [1, m]$	Transformation space
$\varepsilon_E(r_i) = (r_i, f_1(r_i), f_2(r_i), f_3(r_i), \ldots)$	External data function
$E = \{f_j : R \to X_j\}_j$	External data
X_j	External data attribute
$\mathfrak{R}_E$	External data resolution strategy
$\mathfrak{R} : \{r_l, r_m, \ldots\} \overset{\varepsilon_E}{\longmapsto} \{\varepsilon_E(r_l), \varepsilon_E(r_m), \ldots\} \overset{\mathfrak{R}_E}{\longmapsto} a$	Resolution strategy
$r = (c, a)$	Rule
$R = \{r_i\}_i, i \in [1, n]$	Rule set

$\mathfrak{R} : 2^R \rightarrow \mathcal{A}$	Generic resolution strategy
$E = \{E_1, E_2, \ldots\}$	External data set
a_d	Default action
$(R, \mathfrak{R}, \emptyset, a_d)$	Policy
match_R	Matching function
$\mathrm{eff}_p(r)$	Effective Function
$\mathrm{effc}_p(r)$	Effective Cover Function
$r^{\mathrm{T}} = \left(c^{\mathrm{T}}, a\right)$	Transformed rule
$c^{\mathrm{T}} = s_1^{\mathrm{T}} \times s_2^{\mathrm{T}} \times \cdots \times s_m^{\mathrm{T}}$	Transformed condition clause
$\mathrm{in}(\tau_i) = c$	Transformation-condition extraction
$\mathrm{out}(\tau_i) = \gamma_i$	Transformation-action extraction
$\Psi\left(r_i\right) = \prod_{j \in I_{\mathrm{T}}} s_{ij}$	Projection on transformation space
$\mathcal{C}(r_i) = c_i$	Rule condition extraction
$\Theta(r_i, \tau_j) : r_i \longmapsto r_i^{(j)}$	Rule transformation function
$r^{(\tau_j)} = (\Sigma(\mathbb{F}_I, in(\tau_j)), a_d) r^{(\tau_j) \in D^{(T)}}$	Default transformation rule
$D^{(T)}$	Default transformation rule-set
$p^{(T)} = (R \cup R^{(T)}, \mathfrak{T}, E, d)$	T-modified policy
$\mathfrak{T}$	T-modified resolution strategy
$R^{(T)} = \{\Theta(r_i, \tau_j)\}_{i,j} \cup D^{(T)}$, with $i \leq m, j \leq t$	T-modified rule-set
$p^{(T_1, T_2)}$	(T_1, T_2)-modified policy
$R \cup R^{(T_1)} \cup \left(R \cup R^{(T)}\right)^{(T_2)} = R \cup R^{(T_1)} \cup R^{(T_2)} \cup R^{(T_2, T_1)}$	(T_1, T_2)-modified rule-set

$\circ : R \times R \times \cdots \times R \to k^{\mathfrak{S}} \times \mathcal{A}$	Rule composition
$\overline{R}$	Closure
$+ : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$	Serial action composition
$\mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2} : 2^{R_1 \cup R_2}$	Composed resolution strategy
$(R_1 \cup R_2, \mathbb{R}_{+,\mathfrak{R}_1,\mathfrak{R}_2}, \emptyset, a_{d1} + a_{d2})$	Serial composed policy
$\mathrm{CAN} : 2^{R^*} \to \mathcal{A}$	Canonical form
$G(p)$	Cover graph
$d = (D, \mathfrak{R}, E, \mathrm{ID})$	Difference policy
$\sim : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$	Action comparison composition
$\mathbb{R}_{\sim,\mathfrak{R}_1,\mathfrak{R}_2} : 2^{R_1 \cup R_2}$	Comparison resolution strategy
$p_1 \sim p_2 = (R_1 \cup R_2, \mathbb{R}_{\sim,\mathfrak{R}_1,\mathfrak{R}_2}, \emptyset, a_{d1} \sim a_{d2})$	Comparison policy

Bibliography

[1] A.Westerinen, J.Schnizlein, J.Strassner, M.Scherling, "Terminology for Policy-Based Management", RFC-RFC-3198, November 2001.

[2] B.Moore, E.Ellesson, J.Strassner, W.A., "Policy Core Information Model", RFC-RFC-3060, February 2001.

[3] Ponemon Institute, "2014 Global Report on the Cost of Cyber Crime" 2014.

[4] D. A.Patterson, "A simple way to estimate the cost of downtime", LISA2002 : 16th Systems Administration Conference, Usenix, , Philadelphia, US-PE, November 3–8 2002, pp. 185–188.

[5] K. L.Gwebu, J.Wang, W.Xie, "Understanding the Cost associated with Data Security Breaches", PACIS2014 : Pacific Asia Conference on Information Systems, Chengdu, China, June 24–28 2014.

[6] M.Ko, C.Dorantes, "The Impact of Information Security Breaches on Financial Performance of the Breached Firms: an Empirical Investigation", Information Resources Management Journal, Vol. 17, No. 2 2006, pp. 13–22.

[7] K. M.Gatzlaff, K. A.McCullough, "The effect of data breaches on shareholder wealth", Risk Management and Insurance Review, Vol. 13, No. 1, March 2010, pp. 61–83, doi:10.1111/j.1540-6296.2010.01178.x.

[8] J.Goldstein, A.Chernobai, M.Benaroch, "An event study analysis of the economic impact of IT operational risk and its subcategories", Journal of the Association for Information Systems, Vol. 12, No. 9, September 2011, pp. 606–631.

[9] A.Wool, "Firewall Configuration Errors Revisited", CoRR, Vol. abs/0911.1240 2009, pp. 103–122.

[10] W.Avishai, "Trends in Firewall Configuration Errors: Measuring the Holes in Swiss Cheese", IEEE Internet Computing, Vol. 14, No. 4, July 2010, pp. 58–65, doi:10.1109/MIC.2010.29.

[11] Verizon Enterprise, "Data Breach Investigations Report" 2008, www.verizonenterprise.com/resources/security/reports/2009_databreach_rp.pdf.

[12] Verizon Enterprise, "Data Breach Investigations Report" 2014,

www.verizonenterprise.com/DBIR/2014/reports/rp_Verizon-DBIR-2014_en_xg.pdf.

[13] Verizon Enterprise, "Data Breach Investigations Report" 2015, www.verizonenterprise.com/resources/reports/rp_data-breach-investigation-report-2015_en_xg.pdf.

[14] 7Safe Limited, "UK Security Breach Investigations Report" 2010, www.7safe.com/breach_report/Breach_report_2010.pdf.

[15] S.Wright, "PCI DSS V1.2: A Practical Guide to Implementation", It Governance Ltd, 2009, ISBN: 978-1849280235.

[16] Forrester Research, "How To Manage Your Information Security Policy Framework" 2006.

[17] D.Oppenheimer, "The importance of understanding distributed system configuration", CHI03 : Human Factors in Computer Systems, Ft. Lauderdale, US-FL, April 5 2003.

[18] T.European Telecommunications Standards Institute, "Network function virtualization - White Paper 2", October 2013.

[19] N.Feamster, J.Rexford, E.Zegura, "The Road to SDN: An Intellectual History of Programmable Networks", ACM SIGCOMM Computer Communication Review, Vol. 44, No. 2, April 2014, pp. 87–98, doi:10.1145/2602204.2602219.

[20] C.Pitscheider, "Network-Security-Policy Analysis", DEPEND 2014, 7th International Conference on Dependability, Lisbon, Portugal, November 16–20 2014, pp. 10–16.

[21] C.Basile, A.Cappadonia, A.Lioy, "Geometric interpretation of policy specification", POLICY 2008 : IEEE Workshop on Policies for Distributed Systems and Networks, Palisades, NY-US, June 2–4 2008, pp. 78–81, doi:10.1109/POLICY.2008.36.

[22] C.Basile, D.Canavese, A.Lioy, C.Pitscheider, F.Valenza, "Inter-function anomaly analysis for correct SDN/NFV deployment", International Journal of Network Management, Vol. 26, No. 1, January 2016, pp. 25–43, doi:10.1002/nem.1917.

[23] J.Qian, S.Hinrichs, K.Nahrstedt, "ACLA: A Framework for Access Control List (ACL) Analysis and Optimization", CMS01 : 5th Joint Working Conference on Communications and Multimedia Security, Darmstadt, Germany, May 21-22 2001, pp. 197–211, doi:10.1007/978-0-387-35413-2_18.

[24] E.Al-Shaer, H.Hamed, "Firewall Policy Advisor for anomaly discovery and rule editing", IFIP/IEEE 8th International Symposium on Integrated Network Management, Colorado Springs, CO-US, March 24–28 2003, pp. 17–30, doi:10.1109/INM.2003.1194157.

[25] E.Al-Shaer, H.Hamed, "Discovery of policy anomalies in distributed firewalls", INFOCOM2004 : 23rd AnnualJoint Conference of the IEEE Computer and Communications Societies, Hong Kong, Cina, March 7–11 2004, pp. 2605–2616, doi:10.1109/INFCOM.2004.1354680.

[26] E.Al-Shaer, H.Hamed, R.Boutaba, M.Hasan, "Conflict classification and analysis of distributed firewall policies", IEEE Journal on Selected Areas in Communications, Vol. 23, No. 10, October 2005, pp. 2069–2084, doi:10.1109/JSAC.2005.854119.

[27] N.Lehmann, R.Schwarz, J.Keller, "FIRECROCODILE: A Checker for Static Firewall Configurations", SAM06: International Conference on Security & Management, Las Vegas, NV, June 26–29 2006, pp. 193–199, doi:10.1.1.88.3899.

[28] L.Yuan, H.Chen, J.Mai, C.-n.Chuah, "FIREMAN: A Toolkit for FIREwall Modeling and ANalysis", IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA-US, May 21–24 2006, pp. 199–213, doi:10.1109/SP.2006.16.

[29] A.Jeffrey, T.Samak, "Model Checking Firewall Policy Configurations", POLICY2009 : IEEE International Symposium on Policies for Distributed Systems and Networks 2009, London, UK, July 20–22 2009, pp. 60–67, doi:10.1109/POLICY.2009.32.

[30] H.Hu, G.-J.Ahn, K.Kulkarni, "FAME: a firewall anomaly management environment", SafeConfig10 : 3rd ACM workshop on Assurable and usable security configuration, Chicago, IL-US, October 4-8 2010, pp. 17–26, doi:10.1145/1866898.1866902.

[31] H.Hu, G. J.Ahn, K.Kulkarni, "Detecting and resolving firewall policy anomalies", IEEE Transactions on Dependable and Secure Computing, Vol. 9, No. 3, May-June 2012, pp. 318–331, doi:10.1109/TDSC.2012.20.

[32] W.Krombi, M.Erradi, A.Khoumsi, "Automata-Based Approach to Design and Analyze Security Policies", PST2014 : 12th International Conference on Privacy, Security and Trust, Toronto, Canada, July 23-24 2014, pp. 306–313, doi:10.1109/PST.2014.6890953.

[33] V.Capretta, B.Stepien, A.Felty, S.Matwin, "Formal correctness of conflict detection for firewalls", FMSE07 : ACM Workshop on Formal Methods in Security Engineering, Alexandria, VA, October 29 - November 2 2007, pp. 22–30, doi:10.1145/1314436.1314440.

[34] T.Abbes, A.Bouhoula, M.Rusinowitch, "An inference system for detecting firewall filtering rules anomalies", SAC08: ACM Symposium on Applied Computing, Fortaleza, Brazil, March 16–20 2008, pp. 2122–2128, doi:10.1145/1363686.1364197.

[35] N.Basumatary, S. M.Hazarika, "Model checking a firewall for anomalies", ICETACS2013 : 1st International Conference on Emerging Trends and Applications in Computer Science, Shillong, India, September 13-14 2013, pp. 92–96, doi:10.1109/ICETACS.2013.6691402.

[36] K.Golnabi, R. K.Min, L.Khan, E.Al-Shaer, "Analysis of firewall policy rules using data mining techniques", NOMS2006 : 10th IEEE/IFIP Network Operations and Management Symposium, Vancouver, Canada, April 3–7 2006, pp. 305–315, doi:10.1109/NOMS.2006.1687561.

[37] C.Basile, A.Cappadonia, A.Lioy, "Network-Level Access Control Policy Analysis and Transformation", IEEE/ACM Transactions on Networking, Vol. 20, No. 4, August 2012, pp. 985–998, doi:10.1109/TNET.2011.2178431.

[38] J.Garcia-Alfaro, N.Boulahia, F.Cuppens, "Complete analysis of configuration rules to guarantee reliable network security policies", International Journal of Information Security, Vol. 7, No. 2, April 2007, pp. 103–122, doi:10.1007/s10207-007-0045-7.

[39] J.Alfaro, F.Cuppens, N.Cuppens, "Analysis of Policy Anomalies on Distributed Network Security Setups", ESORICS2006 : 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18–20 2006, pp. 496–511, doi:10.1007/11863908_30.

[40] J.Garcia-Alfaro, F.Cuppens, N.Boulahia, P.Stere, "MIRAGE: a management tool for the analysis and deployment of network security policies", SETOP2010: 3rd International Workshop on Autonomous and Spontaneous Security, Athens, Greece, September 23 2011, pp. 203–215, doi:10.1007/978-3-642-19348-4_15.

[41] F.Cuppens, "Handling Stateful Firewall Anomalies", SEC2012: 27th IFIP TC 11 Information Security and Privacy Conference, Heraklion, Greece, June 4–6 2012, pp. 174–186, doi:10.1007/978-3-642-30436-1_15.

[42] J.Garcia-Alfaro, F.Cuppens, N.Boulahia, S.Martinez, J.Cabot, "Management of stateful firewall misconfiguration", Computers & Security, Vol. 39, No. A, November 2013, pp. 64–85, doi:10.1016/j.cose.2013.01.004.

[43] C.Basile, A.Lioy, "Analysis of Application-Layer Filtering Policies With Application to HTTP", IEEE/ACM Transactions on Networking, Vol. 23, No. 1, February 2015, pp. 28 – 41, doi:10.1109/TNET.2013.2293625.

[44] A. X.Liu, M. G.Gouda, "Complete Redundancy Detection in Firewalls", IFIP WG 11.3 : 19th Working Conference on Data and Applications Security, Storrs, CT-US, August 7-10 2005, pp. 193–206, doi:10.1007/11535706_15.

[45] J. G.-A.F. Cuppens, N. Boulahia, "Detection and Removal of Firewall Misconfiguration", IASTED2015 : International Conference on Communication, Network

and Information Security, Phoenix, AZ-US, November 14–16 2005, pp. 42–53, doi:10.1.1.145.2112.

[46] F.Cuppens, N.Boulahia, J.Garcia, "Detection of network security component misconfiguration by rewriting and correlation", PPDP11 : 13th international ACM SIGPLAN symposium on Principles and practices of declarative programming, Odense, Denmark, July 20–22 2006, pp. 77–88, doi:10.1145/2003476.2003489.

[47] M.Abedin, S.Nessa, L.Khan, B.Thuraisingham, "Detection and Resolution of Anomalies in Firewall Policy Rules", 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Sophia Antipolis, France, July 31 – August 2 2006, pp. 15–29, doi:10.1007/11805588_2.

[48] S.Ferraresi, S.Pesic, L.Trazza, A.Baiocchi, "Automatic conflict analysis and resolution of traffic filtering policy for firewall and security gateway", ICC07 : IEEE International Conference on Communications, Glasgow, UK, June 24–28 2007, pp. 1304–1310, doi:10.1109/ICC.2007.220.

[49] R.Oliveira, S.Lee, H.Kim, "Automatic detection of firewall misconfigurations using firewall and network routing policies", PFARM: Workshop on proactive failure avoidance, recovery and maintenance, Lisbon, Portugal, June 29–July 2 2009.

[50] A.Saadaoui, N. B. Y. B.Souayeh, A.Bouhoula, "Formal approach for managing firewall misconfigurations", RCIS2014 : IEEE 8th International Conference on Research Challenges in Information Science, Marrakech, Morocco, May 28-30 2014, pp. 1–10, doi:10.1109/RCIS.2014.6861044.

[51] Z.Fu, S. F.Wu, H.Huang, K.Loh, F.Gong, I.Baldine, C.Xu, "IPSec/VPN Security Policy: Correctness, Conflict Detection, and Resolution", POLICY2001 : International Workshop on Policies for Distributed Systems and Networks, Bristol, UK, January 29–31 2001, pp. 39–56, doi:10.1007/3-540-44569-2_3.

[52] E.Al-Shaer, H.Hamed, W.Marrero, "Modeling and Verification of IPSec and VPN Security Policies", ICNP2005 : 13th IEEE International Conference on Network Protocols, Boston, MA-US, November 6–9 2005, pp. 259–278, doi:10.1109/ICNP.2005.25.

[53] E.Al-Shaer, H.Hamed, "Taxonomy of conflicts in network security policies", IEEE Communications Magazine, Vol. 44, No. 3, March 2006, pp. 134–141, doi:10.1109/MCOM.2006.1607877.

[54] Z.Li, X.Cui, L.Chen, "Analysis And Classification of IPSec Security Policy Conflicts", FCST06: Japan-China Joint Workshop on Frontier of Computer Science and Technology, Fukushimna, Japan, November 17–18 2006, pp. 83–88, doi:10.1109/FCST.2006.10.

[55] S.Niksefat, M.Sabaei, "Efficient Algorithms for Dynamic Detection and Resolution of IPSec/VPN Security Policy Conflicts", AINA2010 : 24th IEEE International Conference on Advanced Information Networking and Applications, Perth, WA, April 20–23 2010, pp. 737–744, doi:10.1109/AINA.2010.99.

[56] F.Valenza, C.Basile, D.Canavese, A.Lioy, "Inter-technology conflict analysis for communication protection policies", CRiSIS2014: 9th International Conference on Risks and Security of Internet and Systems, Trento, Italy, August 27–29 2014, pp. 148–163, doi:10.1007/978-3-319-17127-2_10.

[57] Dan, Farmer and Wietse Venema, "SATAN: Security Administrator's Tool for Analyzing Networks" 1995.

[58] Harry Anderson, "Introduction ti Nessus" 2001.

[59] Andrea Barisani, "Testing firewalls and IDS with FTester" 2001.

[60] A.El-Atawy, T.Samak, Z.Wali, E.Al-Shaer, F.Lin, C.Pham, S.Li, "An Automated Framework for Validating Firewall Policy Enforcement", POLICY07 : 8th IEEE International Workshop on Policies for Distributed Systems and Networks, Bologna, Italy, June 13–15 2007, pp. 151–160, doi:10.1109/POLICY.2007.5.

[61] E.Al-Shaer, A.El-Atawy, T.Samak, "Automated pseudo-live testing of firewall configuration enforcement", IEEE Journal on Selected Areas in Communications, Vol. 27, No. 3, April 2009, pp. 302–314, doi:10.1109/JSAC.2009.090406.

[62] A. D.Brucker, L.BrÃ¼gger, B.Wolff, "hol-TestGen/fw", 10th International Colloquium, Shanghai, China, September 4-6 2013, pp. 112–121, doi:10.1007/978-3-642-39718-9_7.

[63] a.Mayer, a.Wool, E.Ziskind, "Fang: a firewall analysis engine", S&P2000 : IEEE Symposium on Security and Privacy, Berkeley, CA, May 14–17 2000, pp. 177–187, doi:10.1109/SECPRI.2000.848455.

[64] A.Wool, "Architecting the Lumeta Firewall Analyzer", SSYM01 : 10th USENIX Security Symposium, Washington, DC, August 13–17 2001, pp. 85–97.

[65] A.Mayer, A.Wool, E.Ziskind, "Offline firewall analysis", International Journal of Information Security, Vol. 5, No. 3, July 2006, pp. 125–144, doi:10.1007/s10207-005-0074-z.

[66] P.Eronen, J.Zitting, "An expert system for analyzing firewall rules", NordSec2001 : 6th Nordic Workshop on Secure IT Systems, Lyngby, Denmark 2001, pp. 100–107, doi:10.1.1.21.4430.

[67] S.Hazelhurst, "Algorithms for Verifying Firewall and Router Access Lists", 2003 46th Midwest Symposium on Circuits and Systems, Cairo, Egypt, December 27–30 2003, pp. 512–515, doi:10.1109/MWSCAS.2003.1562330.

[68] G.Xie, D.Maltz, A.Greenberg, G.Hjalmtysson, J.Rexford, "On static reachability analysis of IP networks", INFOCOM2005: 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL-US, March 13–17 2005, pp. 2170–2183, doi:10.1109/INFCOM.2005.1498492.

[69] S.Bandhakavi, S.Bhatt, C.Okita, P.Rao, "Analyzing end-to-end network reachability", IM09: IFIP/IEEE International Symposium on Integrated Network Management, Long Island, NY-US, June 1–5 2009, pp. 585–590, doi:10.1109/INM.2009.5188865.

[70] R.Marmorstein, P.Kearns, "A Tool for Automated iptables Firewall Analysis", ATEC05 : USENIX Annual Technical Conference, Anaheim, CA-US, April 10–15 2005, pp. 71–81, doi:10.1.1.99.1191.

[71] R.Marmorstein, P.Kearns, "An open source solution for testing NAT'd and nested iptables firewalls", LISA05 : 19th Large Installation Systems Administration Conference, San Diego, CA-US, December 4–9 2005, pp. 103–112, doi:10.1.1.436.5766.

[72] R.Marmorstein, P.Kearns, "Debugging a firewall policy with policy mapping", ;login:, Vol. 32, No. 1, February 2007, pp. 44–51.

[73] P.Matoušek, J.Ráb, O.Ryšavý, M.Švéda, "A formal model for network-wide security analysis", ECBS2008 : 15th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, Belfast, UK, March 31 – April 4 2008, pp. 171–181, doi:10.1109/ECBS.2008.13.

[74] M.Christiansen, E.Fleury, M.Christiansen, E.Fleury, "An Interval Decision Diagram Based Firewall", ICN04 : 3th IEEE International Conference on Networking, Point-Ã -Pitre, Guadeloupe, March 1–4 2004.

[75] A.Khakpour, A.Liu, "Quarnet: A Tool for Quantifying Static Network Reachability", IEEE/ACM Trans. Netw., Vol. 21, No. 2, February 2009, pp. 551 – 565.

[76] T.Nelson, D. J.Dougherty, C.Barratt, K.Fisler, "The Margrave Tool for Firewall Analysis", LISA10 : 24th USENIX Conference on Large Installation System Administration, San Jose, CA-US, November 7–12 2010.

[77] H.Mai, A.Khurshid, R.Agarwal, M.Caesar, P. B.Godfrey, S. T.King, "Debugging the data plane with anteater", SIGCOMM11 : ACM Special Interest Group on Data Communication, Toronto, Canada, August 15–19 2011, pp. 290–301, doi:10.1145/2043164.2018470.

[78] E.Al-Shaer, M. N.Alsaleh, "ConfigChecker: A tool for comprehensive security configuration analytics", SAFECONFIG2011 : 4th Symposium on Configuration Analytics and Automation, Arlington, US-VA, October 31 - November 1 2011, pp. 1–2, doi:10.1109/SafeConfig.2011.6111667.

[79] M.Sveda, O.Rysavy, G. D.Silva, "Static Analysis of Routing and Firewall Policy Configurations", ICETE10: 7th International Joint Conference, Athens, Greece, July 26–28 2010, pp. 39–53, doi:10.1007/978-3-642-25206-8_2.

[80] P.Kazemian, G.Varghese, N.McKeown, "Header space analysis: Static checking for networks", NSDI12: 9th USENIX conference on Networked Systems Design and Implementation, San Jose, CA, April 25–27 2012, pp. 9–9.

[81] J.Guttman, "Filtering postures: Local enforcement for global policies", IEEE Symposium on Security and Privacy 1997, Oakland, CA-US, May 4–7 1997, pp. 120 – 129, doi:10.1109/SECPRI.1997.601327.

[82] J. D.Guttman, A. L.Herzog, "Rigorous automated network security management", International Journal of Information Security, Vol. 4, No. 1-2, February 2005, pp. 29–48, doi:10.1007/s10207-004-0052-x.

[83] A.Liu, "Formal Verification of Firewall Policies", ICC2008 : IEEE International Conference on Communications, Beijing, China, May 19–23 2008, pp. 1494–1498, doi:10.1109/ICC.2008.289.

[84] A.Liu, M.Gouda, "Diverse Firewall Design", DSN04 : International Conference on Dependable Systems and Networks, Florence, Italy, June 28 – July 1 2004, pp. 595–604, doi:10.1109/DSN.2004.1311930.

[85] A.Liu, M.Gouda, "Diverse Firewall Design", IEEE Transactions on Parallel and Distributed Systems, Vol. 19, No. 9, September 2008, pp. 1237–1251, doi:10.1109/TPDS.2007.70802.

[86] Y.Yin, R.Bhuvaneswaran, "Inferring the Impact of Firewall Policy Changes by Analyzing Spatial Relations between Packet Filters", ICCT06: International Conference on Communication Technology, Guilin, China, November 27–30 2006, pp. 1–6, doi:10.1109/ICCT.2006.341930.

[87] A.Liu, "Change-impact analysis of firewall policies", 12th European Symposium On Research In Computer Security, Dresden, Germany, September 24–26 2007, pp. 155–170, doi:10.1007/978-3-540-74835-9_11.

[88] A.Liu, "Firewall policy change-impact analysis", ACM Transactions on Internet Technology, Vol. 11, No. 4, March 2012, pp. 1–24, doi:10.1145/2109211.2109212.

[89] N.Ben Youssef, A.Bouhoula, F.Jacquemard, "Automatic verification of conformance of firewall configurations to security policies", ISCC2009 : 20th IEEE Symposium on Computers and Communications, Sousse, Tunisia, July 5–8 2009, pp. 526–531, doi:10.1109/ISCC.2009.5202309.

[90] N. B.Youssef, A.Bouhoula, "Dealing with Stateful Firewall Checking", DICTAP2011 : International Conference on Digital Information and Communication Technology and its Applications, Dijon, France, June 21–23 2011, pp. 493–507,

doi:10.1007/978-3-642-21984-9_42.

[91] C.Basile, D.Canavese, A.Lioy, C.Pitscheider, "Improved reachability analysis for security management", PDP2013 : 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing, Belfast, UK, February 27 – March 1 2013, pp. 534–541, doi:10.1109/PDP.2013.86.

[92] J.van Lunteren, T.Engbersen, "Fast and scalable packet classification", IEEE Journal on Selected Areas in Communications, Vol. 21, No. 4, May 2003, pp. 560–571, doi:10.1109/JSAC.2003.810527.

[93] J. E.Hopcroft, R.Motwani, J. D.Ullman, "Introduction to Automata Theory, Languages, and Computation", Addison Wesley, 1979, ISBN: 978-0321455369.

[94] P.Linz, "An Introduction to Formal Languages and Automata", Jones & Bartlett Learning, 2001, ISBN: 978-1449615529.

[95] J. A.Brzozowski, "Derivatives of Regular Expressions", Journal of the ACM, Vol. 11, No. 4, October 1964, pp. 481–494, doi:10.1145/321239.321249.

[96] S.Kleene, "Representation of Events in Nerve Nets and Finite Automata" 1951.

[97] A. X.Liu, M. G.Gouda, H.Ma, A. H.Ngu, "Firewall Queries", OPODIS2004 : 8th International Conference on Principles of Distributed Systems, Grenoble, France, December 15–17 2004, pp. 124–139, doi:10.1007/11516798_15.

[98] AlgoSec Inc., "Examining the Dangers of Complexity in Network Security Environments: AlgoSec Survey Insights" 2012, www.algosec.com/resources/files/Specials/Survey%20files/12_10_11_security_complexity.pdf.

[99] D.Taylor, "Survey and taxonomy of packet classification techniques", ACM Computing Surveys, Vol. 37, No. 3, September 2005, pp. 238–275, doi:10.1145/1108956.1108958.

[100] E.Al-Shaer, H.Hamed, "Modeling and Management of Firewall Policies", IEEE Transactions on Network and Service Management, Vol. 1, No. 1, April 2004, pp. 2–10, doi:10.1109/TNSM.2004.4623689.